Howard Dachslager
Masato Hayashi
Richard Zucker

Programación en BASIC

Un método práctico

PROGRAMACION EN BASIC

Un método práctico

Programación en BASIC

Un método práctico

Howard Dachslager
Masato Hayashi
Richard Zucker

ANAYA MULTIMEDIA

MICROINFORMATICA

Título de la obra original:
«LEARNING BASIC PROGRAMMING»

Traducción de: Javier David García.

Diseño de colección: Antonio Lax.

Reservados todos los derechos. Ni la totalidad ni parte de este libro puede reproducirse o transmitirse por ningún procedimiento electrónico o mecánico, incluyendo fotocopia, grabación magnética o cualquier almacenamiento de información y sistema de recuperación, sin permiso escrito de Ediciones Anaya-Multimedia, S. A.

ORIGINAL ENGLISH LANGUAGE EDITION PUBLISHED BY
Brooks/Cole Publishing Company, a Division of
Wadsworth, Inc., 555 Abrego Street, Monterey,
CA 93940 U.S.A.
COPYRIGHT 1983 by WADSWORTH, Inc., 10 Davis Street,
Belmont, CA 94002 U.S.A.
ALL RIGHTS RESERVED

© EDICIONES ANAYA MULTIMEDIA, S. A., 1984
Villafranca, 22. 28028 Madrid
Depósito legal: M. 29.624-1984
I.S.B.N.: 84-7644-000-2
Printed in Spain
Imprime: Gráf. FUTURA, Sdad. Coop. Ltda.
Villafranca del Bierzo, 21-23. FUENLABRADA (Madrid)

Prólogo

Nuestro convencimiento de la falta de un método simple y sistemático para la enseñanza del BASIC fue la motivación para escribir este libro. A través de los años, hemos experimentado cierta frustración en nuestros intentos de enseñar a los estudiantes a escribir programas más allá del nivel elemental. Esto nos llevó a reexaminar los métodos tradicionales de enseñanza de programación y a buscar un nuevo enfoque para el tema. Este libro es el resultado de nuestros esfuerzos.

Teniendo en cuenta el incremento de la popularidad de los microordenadores, hemos decidido utilizar el Applesoft II BASIC, que es el lenguaje estándar del ordenador Apple II Plus. Nuestra intención es, de todas formas, presentar tantos aspectos universales del BASIC como sea posible. Con esto en mente, las características del Applesoft que son estrictamente únicas del Apple II Plus se han dejado para el último capítulo. El resto es la esencia del Applesoft, que es similar a las versiones del BASIC que se pueden encontrar en otros microordenadores.

Cuatro características de este libro lo hacen distinto de otros sobre la programación BASIC, como son los diagramas de cajas, los ejercicios prácticos con el ordenador, estudios de algoritmos y módulos de programa, y un capítulo entero dedicado a los «extras» encontrados únicamente en el Applesoft BASIC. A continuación explicamos brevemente estas técnicas.

El diagrama de cajas

El diagrama de cajas es como una ventana en la memoria del ordenador. Los estudiantes pueden ver cómo son almacenadas las variables y cómo cambian los datos. Las ideas normalmente abstractas sobre almacenamiento de datos y secuenciamiento de programas son visualizadas, de tal manera que el aprendizaje se convierte en conceptual más que en memorístico. Un buen ejemplo de las cualidades de los diagramas de cajas se puede encontrar en las páginas 25 y 26, donde se utilizan para demostrar la técnica del intercambio de contenidos de dos variables. Los diagramas de cajas son utilizados a lo largo del libro, haciendo visibles los efectos que las instrucciones BASIC producen en las variables.

Ejercicios prácticos

Al final de cada uno de los diez primeros capítulos hay ejercicios que tratan sobre los puntos más importantes de cada uno. Estos ejercicios están pensados para realizarlos directamente en un ordenador. Creemos que la experiencia real con un ordenador acelera el aprendizaje. Cada ejercicio es una actividad estructurada con la forma «haga esto y aquello», con preguntas que fuerzan al estudiante a analizar las respuestas del ordenador. Las respuestas y explicaciones se dan inmediatamente después de las preguntas de los ejercicios, de tal manera que los estudiantes nunca están sin saber la respuesta demasiado tiempo. Los ejercicios han supuesto una gran ayuda para nuestros estudiantes. Muchos de ellos nos han dicho que los ejercicios han eliminado su miedo al ordenador y han aumentado su comprensión del BASIC.

Algoritmos y módulos

Creemos que una temprana introducción a la escritura de programas utilizando módulos es importante para el buen desarrollo de las habilidades de programación. En el capítulo 3 tratamos el método ACS (en inglés, ACO: Assignment, Compute, Output-Asignación, Cómputo, Salida), que da al estudiante una forma de estructurar programas en módulos simples. Más adelante volvemos a tratar el tema introduciendo la noción de algoritmo. Los estudiantes aprenden a unir algoritmos para formar módulos. Los programas nacen a partir de estos módulos (ver capítulo 10). Esta técnica ha ayudado a nuestros estudiantes a «dividir y vencer» a programas largos, partiéndolos en otros más pequeños. El sistema de módulos también crea buenos hábitos de programación.

Características especiales del Applesoft BASIC

El último capítulo presenta algunas características interesantes del lenguaje Applesoft. Es una bien merecida recompensa para el estudiante que ha logrado alcanzar el final del libro. Aquí se encuentran temas como los *peeks* y los *pokes*, los métodos de impresión de datos, gráficas en alta y baja resolución, manejo del altavoz, mandos de control de juegos *(paddles)* y mucho más. Creemos que el lector quedará complacido con estos temas y, en nuestra opinión, conseguirá una buena idea de la gran potencia del Apple II Plus.

Notas para el profesor

Le remitimos al índice de contenidos para conocer la lista completa de temas y el orden en que aparecen. No obstante, le ofrecemos los siguientes puntos para su consideración:

1 El estudio del BASIC en el capítulo 1 está hecho en modo inmediato. Esto permite al estudiante ir directamente al computador sin haber tenido que estudiar conceptos más difíciles relacionados con el modo diferido. Utilizando las sentencias LET y PRINT solamente, los estudiantes aprenden sobre variables, asignaciones y aritmética, evitando la tentación de pasar ligeramente sobre estos temas. Una buena comprensión de este capítulo será indispensable cuando se expongan conceptos más difíciles.

2 El capítulo 2 trata de las funciones BASIC. Un tratamiento tan temprano de las funciones es romper con las tradiciones, pero creemos que los estudiantes deben trabajar con las funciones tan pronto como sea posible, de manera que comprendan el poder interno del ordenador. Este capítulo puede saltarse, si el instructor da las funciones como tema sabido.

3 Los diagramas de cajas son utilizados a lo largo de todo el libro, pero son especialmente útiles en los capítulos 6, 7 y 8 donde se tratan las instrucciones IF-THEN y FOR-NEXT y las matrices. En el capítulo 6, los estudiantes pueden «ver» el resultado de una sentencia de decisión. En el capítulo 7, con el tema de bucles (que resulta particularmente difícil a los estudiantes) los diagramas de cajas exponen claramente cómo el índice y otras variables cambian durante el proceso del bucle. En el capítulo 8, los diagramas de cajas son la técnica natural para tratar el concepto de matriz y de las variables indexadas.

4 En el capítulo 9 se trata el tema de la concatenación. En el apartado 9.3 se explica cómo la concatenación se utiliza para crear registros a partir de campos de una longitud prefijada. La importancia de esta sección no puede ser subestimada. El tema de los campos y de los registros juega un papel principal en los capítulos 10 y 11, eso sin tener en cuenta las importantes aplicaciones en la «vida real».

5 El capítulo 10 trata la programación estructurada como, por ejemplo, el uso de algoritmos para la construcción de módulos, y, a su vez, el uso de éstos para la construcción de programas, subrutinas y la noción de menú. Es recomendable utilizar el tiempo necesario en este capítulo para hacer comprender al estudiante que programar bien significa mucho más que escribir programas que simplemente funcionen.

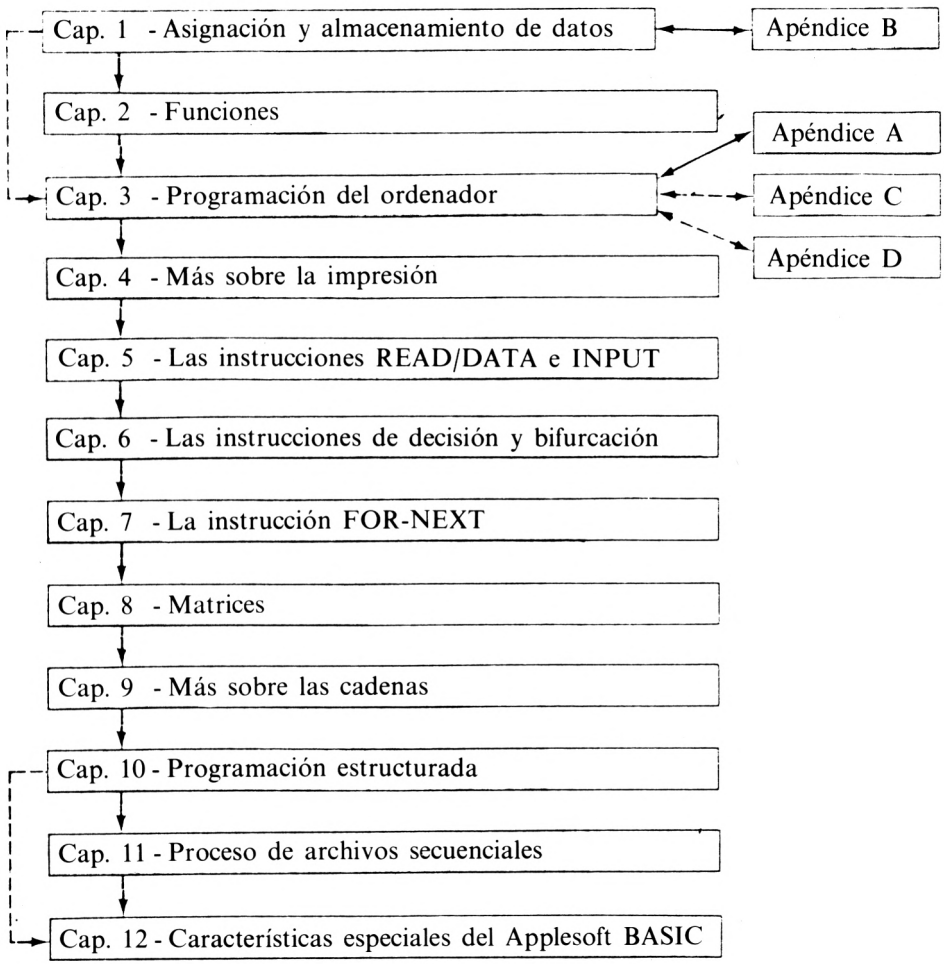
6 La lectura y la escritura de archivos de datos de acceso secuencial o aleatorio no siempre es tema para un primer curso de programación BASIC. El capítulo 11 trata este tema de una manera conservadora, ya que sólo estudia los archivos secuenciales. Este capítulo se puede saltar sin pérdida de continuidad, si se anda escaso de tiempo.

7 En todos los capítulos, excepto en el 12, se ha incluido un apartado denominado «Proyectos». Han sido diseñados para aumentar la comprensión del material de cada capítulo por parte de los estudiantes y para darles aplicaciones adicionales de los temas estudiados.

8 Los apéndices no deben ser vistos por encima. Su relación con los capítulos se muestra en el esquema que sigue a continuación.

Secuencia de capítulos

La línea de puntos indica un camino opcional.



Agradecimientos

Nos gustaría dar las gracias a aquellas personas que ayudaron a hacer realidad este libro.

Gracias por el valioso análisis crítico y las sugerencias de la profesora Susan Anderson de la Universidad Wesleyan de Illinois, del profesor David Hansen del Monterey Peninsula College, del profesor Leon Pearce de la

Universidad de Drake, al doctor Tom Richard de la Universidad del Estado de Bemidji, del profesor Kenneth Trester del Golden West College.

Gracias especiales a Mr. Gerard Walsh por sus muchas horas sobre el ordenador comprobando los errores de los programas ejemplo.

Gracias también a Mr. Gordon McDonald por sus interesantes proyectos, a Mr. Albert Murtz por su asesoramiento en la edición del libro, a Ms. Maureen Conway quien, soportando interminables fines de semana fue, para nosotros, una invalorable ayuda, y a nuestros estudiantes que nos aguantaron.

Por último, pero no por eso menos, a nuestras amadas esposas sin cuya paciencia y apoyo este libro nunca podría haber sido escrito.

HOWARD DACHSLAGER
MASATO HAYASHI
RICHARD ZUCKER

Saddleback College, North Campus
5500 Irvine Center Dr.
Irvine, California 92714

Indice

Prólogo ..... *Pág.* 5

Capítulo 1: Asignación y almacenamiento de datos

1.1. Los datos 17
1.2. El diagrama de cajas 18
1.3. Variables numéricas 19
1.4. Variables tipo cadena 20
1.5. La instrucción LET 21
1.6. Intercambio del contenido de las variables 25
1.7. La instrucción PRINT 29
1.8. Haciendo aritmética 31
1.9. Proyectos 37

Práctica 1.1. La instrucción LET 39
Práctica 1.2. La instrucción PRINT 40
Práctica 1.3. Variables numéricas y de cadena 42
Práctica 1.4. Intercambio 44
Práctica 1.5. Las operaciones aritméticas 46

Capítulo 2: Funciones

2.0. Introducción 49
2.1. Las funciones aritméticas 50

2.2. Las funciones con cadenas 57
2.3. Proyectos 61

Práctica 2.1. La función INT 62
Práctica 2.2. La función RND 64
Práctica 2.3. La función LEN 67
Práctica 2.4. La función VAL 68
Práctica 2.5. La función STR$ 69
Práctica 2.6. La función MID$ 70
Práctica 2.7. La función RIGHT$ 71
Práctica 2.8. La función LEFT$ 72

Capítulo 3: Programación del ordenador

3.0. Introducción 75
3.1. Las líneas numeradas y la orden RUN 76
3.2. La instrucción END 78
3.3. La orden LIST y la edición 78
3.4. Las órdenes NEW y CLEAR 80
3.5. La instrucción REMark 85
3.6. Instrucciones múltiples en una sola línea 86
3.7. Programación estructurada: método de módulos ACS 87
3.8. Proyectos 89

Práctica 3.1. Los modos inmediato y diferido 89
Práctica 3.2. LIST y DEL 91
Práctica 3.3. Estructuración de programas utilizando el método ACS 93
Práctica 3.4. Ordenes de disco 97

Capítulo 4: Más sobre la impresión

4.0. Introducción 101
4.1. La coma 101
4.2. El punto y la coma 103
4.3. La función tabulador, TAB(X) 106
4.4. Proyectos 107

Práctica 4.1. El uso de comas en la instrucción PRINT 109
Práctica 4.2. El uso del punto y coma en la instrucción PRINT 111
Práctica 4.3. La función TAB 112

Capítulo 5: Las instrucciones READ/DATA e INPUT

5.0. Introducción 115
5.1. La instrucción READ/DATA 115
5.2. Suma reiterativa 119

5.3. La instrucción RESTORE 121
5.4. La instrucción INPUT 122
5.5. Proyectos 125

Práctica 5.1. Las instrucciones READ/DATA 126
Práctica 5.2. RESTORE 129
Práctica 5.3. INPUT 130

Capítulo 6: Las instrucciones de decisión y ramificación

6.1. La instrucción IF-THEN 135
6.2. La sentencia GOTO 142
6.3. Proyectos 147

Práctica 6.1. La instrucción IF-THEN 149
Práctica 6.2. Ramificación 150

Capítulo 7: La instrucción FOR-NEXT

7.0. Introducción 153
7.1. Comprensión de la instrucción FOR-NEXT 154
7.2. Aplicaciones 163
7.3. Las instrucciones FOR-NEXT anidadas 167
7.4. Proyectos 168

Práctica 7.1. La instrucción FOR-NEXT 170
Práctica 7.2. Lazos anidados 172

Capítulo 8: Matrices

8.0. Introducción 175
8.1. Matrices numéricas unidimensionales 176
8.2. La instrucción DIMensión 179
8.3. Ordenamiento 183
8.4. Aplicaciones 187
8.5. Matrices numéricas bidimensionales 192
8.6. Matrices de cadena uni y bidimensionales 197
8.7. Proyectos 203

Práctica 8.1. Matrices unidimensionales 205
Práctica 8.2. Las matrices bidimensionales 208

Capítulo 9: Más sobre cadenas

9.0. Introducción 213
9.1. Concatenación 213

9.2. Escritura de números en pesetas y céntimos 215
9.3. Campos y registros 217
9.4. Proyectos 221

Práctica 9.1. Concatenación 222
Práctica 9.2. Ajustado y rellenado 224

Capítulo 10: Programación estructurada

10.0. Introducción 229
10.1. Algoritmos 229
10.2. Módulos y programas 241
10.3. Las sentencias GOSUB y RETURN y las subrutinas 250
10.4. ON-GOTO, ON-GOSUB, y el menú 256
10.5. Programación dedutiva («top-down») 258
10.6. Documentación 259
10.7. Proyectos 260

Práctica 10.1. GOSUB/RETURN 262
Práctica 10.2. Las sentencias ON-GOTO y ON-GOSUB 263

Capítulo 11: Procesamiento de archivos secuenciales

11.0. Introducción 269
11.1. Escritura de registros en un archivo secuencial 269
11.2. Lectura de registros en un archivo secuencial 273
11.3. Almacenamiento de datos por campos 275
11.4. Incorporación de registros a un archivo 277
11.5. Proyectos 278

Capítulo 12: Características especiales del BASIC APPLESOFT

12.0. Introducción 281
12.1. PEEK, POKE y CALL 281
12.2. Estética en la escritura 286
12.3. Gráficas 295
12.4. Control de los mandos exteriores y el altavoz 302
12.5. Miscelánea 307

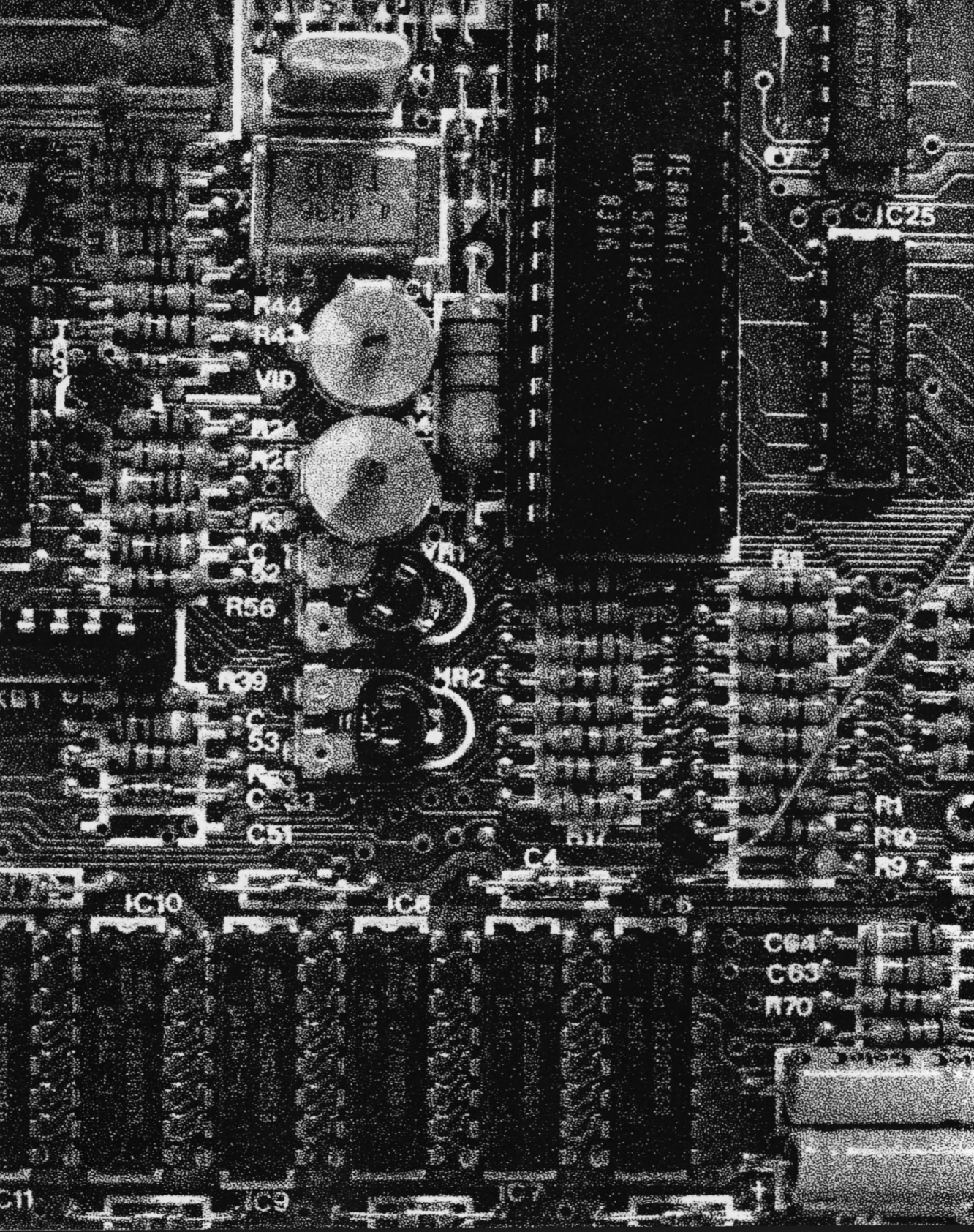
Apéndice A: Cómo opera el APPLE II PLUS 317

Apéndice B: Palabras reservadas en APPLESOFT 321

Apéndice C: Depuración de programas con y sin mensajes de error 323

Apéndice D: Edición de líneas de programa utilizando el control del cursor 329

Solución a los ejercicios de los capítulos 337


IC25
R44
VID
R56
R39
C51
C4
R1
R9
IC10
IC8
C64
C63
R70
IC9
IC7

1

Asignación y almacenamiento de datos

1.1

Los datos

Los ordenadores trabajan con datos. Hay dos tipos de datos usados en BASIC: *numéricos* y *de cadena.*

Los datos numéricos son los números decimales típicos, que incluyen punto decimal y signo negativo cuando es necesario (los datos numéricos pueden ser expresados también con notación científica, como podemos ver en los ejemplos en los ejercicios 1.2). El signo (+) es opcional cuando se expresan números positivos.

Ejemplos de datos numéricos 2.38 5 92714 −0.567 +406.5 0.06

Una *cadena* es cualquier conjunto de letras, dígitos, espacios y símbolos, a los que se denomina simplemente "caracteres"*.

* En algunas versiones del BASIC, el signo de las comillas (") no está permitido como carácter dentro de una cadena. El Applesoft BASIC permite las comillas, si bien se aplican con ciertas restricciones, tal como veremos más adelante.

Ejemplos de cadenas	1OSNE1?	ME GUSTA LA LECHE	HI HO SILVER, AWAY!
	92714	!#$! &%#*#!	UN ANGULO "AGUDO"

Los datos tales como 92714 pueden ser almacenados en la memoria del ordenador bien como un número, bien como una cadena. Cuando 92714 es almacenado como dato numérico, puede ser utilizado para efectuar cálculos. Como cadena encaja mejor en aplicaciones que no implican operaciones artiméticas. Por ejemplo, si 92714 es la población de Irvine, California, el almacenamiento de este número como dato numérico nos permite utilizarlo en los cálculos del censo. Pero cuando 92714 es usado como el *zip code* (código postal) de Irvine, es almacenado como un dato cadena y puede usarse en las aplicaciones propias de las cadenas, tal como en la impresión de rótulos para correspondencia.

1.1

Ejercicios

1. ¿Cuáles de estas expresiones son datos numéricos en BASIC?

   a) 2.34 *b*) 2+3 *c*) −56.43

   d) CALLE DELICIAS 23 *e*) $3.45 *f*) +0.08

2. ¿Cuáles de la siguientes expresiones son cadenas?

   a) −2.34 *b*) $45.00 *c*) UNA RETIRADA A TIEMPO

   d) LA PALABRA "ENIGMA" ES MUY MISTERIOSA

   e) "CORRE MAS DEPRISA"! DIJO RAPIDAMENTE

   f) ES UNA VICTORIA ESTRATEGICA

3. Contar el número de caracteres en las siguientes cadenas:

   a) LA LLUVIA DE ESPAÑA *b*) HI HO SILVER, AWAY!

   c) −23.34 + 47.93 *d*) ANTISISTEMARIANISMO

1.2

El diagrama de cajas

Los datos han de ser almacenados en la memoria del ordenador. Una manera simple para visualizar esto es suponer que el ordenador coloca los datos dentro de cajas de memoria designadas con nombres especiales, tales como X1, Z$, G3 y Y4$.

Este símbolo de caja [ ] representa una posición de memoria donde se pueden almacenar datos.

Ejemplos de diagrama de cajas

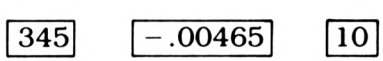
Nos referimos a los datos dentro de una caja como el valor o contenido de la caja. Los nombres de estas cajas se llaman variables. Hay dos tipos de cajas: las que contienen datos numéricos y las que contienen datos tipo cadena, conocidas respectivamente como *variables numéricas* y *variables tipo cadena.*

1.3

Variables numéricas

Las variables numéricas sólo pueden contener datos numéricos.

Ejemplos



Los nombres de las variables numéricas se generan con las letras de la A a la Z y los dígitos del 0 al 9 bajo las siguientes normas.

NORMAS PARA LOS NOMBRES
DE LAS VARIABLES NUMERICAS

1. Todas las variables numéricas deben comenzar con una letra. Esta puede ser seguida por más letras o dígitos hasta un total de 237.
2. Sólo son reconocidos por el ordenador los dos primeros caracteres del nombre de la variable. Por eso, AMILANAR y AMBIDIESTRO nombran a la misma caja en la memoria del ordenador.
3. Ningún nombre de variable numérica puede contener una "palabra reservada", que es una palabra que el ordenador utiliza de una manera especial. Las palabras TO, AND, FOR y AT son palabras reservadas. Así, TOTAL, SANDY, FORTUNA y BEATA no son válidas como nombres de variables, porque cada uno de ellos contiene una palabra reservada. En el apéndice B aparece una lista completa de palabras reservadas Applesoft.

Estas normas se aplican al Applesoft BASIC. En muchas versiones del BASIC, los nombres de variables numéricas deben ser una letra o una

única letra seguida por un dígito solamente. Los nombres de variables largos, como los del Applesoft, permiten a las variables ser descriptivas, tal como SALARIO y PRECIO en vez de S y P.

Ejemplos de nombres de variables numéricas válidas

X X1 GOLF K9 X15 SUMA

Ejemplos de nombres de variables numéricas no válidas

4U	(no puede comenzar por un dígito)
RAZON	(contiene una palabra reservada)
Archivo#3	("#" no es una letra ni un dígito)
D$	("$" no es una letra ni un dígito)

Ejemplo

El diagrama de cajas de la derecha no es correcto, ya que X es una variable numérica y sólo puede contener datos numéricos.

1.3

Ejercicios

1. ¿Cuáles de los diagramas de cajas siguientes son posibles?

2. ¿Cuáles de los nombres de variables numéricas siguientes son correctos?

a) YU *b*) G$ *c*) T45 *d*) RO *e*) TOTAL

f) GOLPEA2ME *g*) 7ANEJO *h*) TASAS

1.4

Variables tipo cadena

Las variables tipo cadena sólo pueden contener cadenas. Una variable tipo cadena en el Applesoft no puede contener ninguna cadena de longitud superior a 255 caracteres (el número de caracteres que puede contener una variable tipo cadena depende de la versión BASIC que se esté utilizando).

Ejemplos

Y$	Q$	T$
HOLA	SILVER	23.6

Los nombres de variables de cadena se crean con letras, dígitos y el símbolo $, bajo la única norma siguiente:

NORMA PARA LOS NOMBRES DE VARIABLES DE CADENA

Todos los nombres de variables tipo cadena se forman poniendo el símbolo $ al final de cualquier nombre válido de variable numérica. Así, IDENTIDAD$ es una variable de cadena válida porque IDENTIDAD es una variable numérica válida.

1.4

Ejercicios

1. ¿Cuáles de los diagramas de cajas son posibles?

a) ZIP$ | 92714 |

b) K9$ | GERMAN SHEPHERD |

c) M$ | SI |

d) GATO$ | TABBY |

e) A$ | TOTAL |

2. ¿Cuáles de estos nombres de variable tipo cadena son válidos?

a) Q$ *b)* TT$ *c)* ZT$ *d)* "Y$" *e)* FORMA$

f) DOMICILIO$ *g)* ZIPCODE

1.5

La instrucción LET

LET es una instrucción que asigna valores a las variables numéricas y tipo cadena. Es uno de los cuatro tipos de las llamadas instrucciones de "asignación".

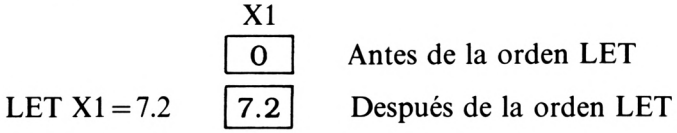
Ejemplos

```
LET X=2
LET Y1=456
LET Q$="ROMEO OH ROMEO"
```

Al ejecutar la orden LET

```
LET X1=7.2
```

el valor numérico 7.2 se coloca dentro de la caja llamada X1. El diagrama de cajas es el siguiente:

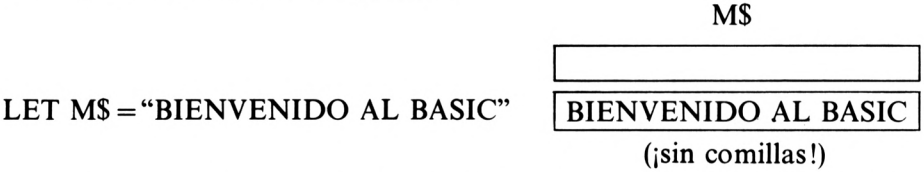


Observemos que la caja llamada X1 tiene un cero dentro de ella antes de la ejecución de la orden LET. Esto ocurre en la mayoría de las versiones del BASIC. Sin embargo, cuando es necesario que la variable contenga un cero, nunca hay que suponer que el cero ya está allí. Hay que asignar el cero a la variable con una instrucción LET.

Al ejecutar la instrucción LET:

LET M$ = "BIENVENIDO AL BASIC"

la cadena BIENVENIDO AL BASIC se coloca en la caja llamada M$. El diagrama de cajas es el siguiente:



Notemos que la caja llamada M$ está en blanco antes de que la orden LET sea dada.

NORMAS IMPORTANTES ACERCA DE LA ASIGNACION DE DATOS TIPO CADENA

1. Siempre que una cadena es asignada a una variable tipo cadena utilizando la instrucción LET, la cadena debe ir entre comillas. Las comillas no se consideran parte de la cadena y no son almacenadas.
2. Puesto que las comillas se usan para encerrar la cadena, no se permiten dentro de ella.

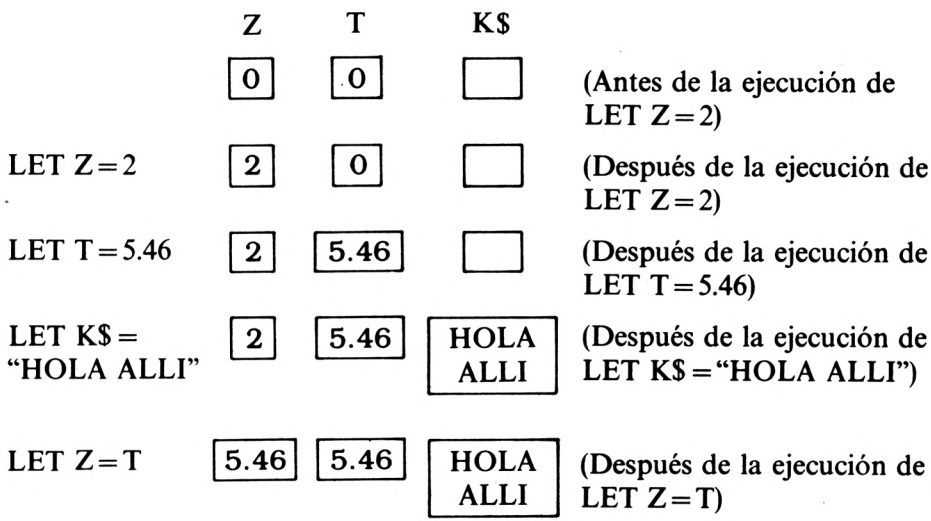
Ejemplo

Correcto	Incorrecto
LET X$ = "BASIC"	LET X$ = BASIC
LET X$ = "UN ANGULO 'AGUDO'"	LET X$ = "UN ANGULO "AGUDO""

Suponga que las siguientes órdenes son introducidas en el ordenador:

```
LET Z=2
LET T=5.46
LET K$="HOLA ALLI"
LET Z=T
```

Los diagramas de cajas para estas instrucciones son los siguientes:



EXPLICACION

Después de la ejecución de la primera instrucción, LET Z=2, el número 2 se colocará en la caja llamada Z.

Después de la segunda instrucción, LET T=5.46, el número 5.46 se colocará en la caja llamada T.

Después de la tercera instrucción, LET K$="HOLA ALLI", la cadena HOLA ALLI se colocará en la caja K$.

Finalmente, después de la última instrucción, LET Z=T, el número que está dentro de la caja T (5.46) será colocado dentro de la caja Z.

Importante: La LET Z=T destruyó el valor 2 dentro de Z durante la transferencia, pero conservó el valor 5.46 en la caja T.

Todos los aspectos importantes de la instrucción LET son discutidos en las cinco reglas siguientes que hemos de conocer bien.

CINCO NORMAS PARA LA INSTRUCCION LET

1.ª Una variable numérica o tipo cadena debe siempre aparecer a la *izquierda* del signo "=".

Ejemplo LET X1=34.5
LET Q5$="DONDE HAS ESTADO ESTA NOCHE?"

2.ª La asignación siempre toma lugar de *derecha a izquierda* a través del signo "=". Cualquier valor que esté a la derecha del signo "=" es colocado dentro de la caja nombrada a la izquierda de éste.

3.ª Los datos transferidos a una variable que esté a la izquierda *siempre* destruyen el dato previamente asignado a esa variable. Después de que un valor es asignado a una variable, este permanece hasta que se asigna un nuevo valor a dicha variable (o hasta que una orden NEW o CLEAR es ejecutada. Consultar el capítulo 3).

Ejemplo

	X	Q$
	0	
LET X=2	2	
LET X=−67.89	−67.89	
LET Q$="HOLA"	−67.89	HOLA
LET Q$="LO"	−67.89	LO

4.ª Si una variable está en el lado derecho del signo "=" la instrucción LET transfiere una *copia* del valor de esa variable a la variable que está en el lado izquierdo.

Ejemplos

	Q	K		X$	Y$
	0	0			
LET Q=6	6	0	LET X$="YO"	YO	
LET K=−7.2	6	−7.2	LET Y$="MIO"	YO	MIO
LET K=Q	6	6	LET X$=Y$	MIO	MIO

5.ª A una variable numérica sólo se le pueden asignar datos numéricos. A una variable tipo cadena sólo se le pueden asignar datos tipo cadena, que deben ir entre comillas, aunque éstas no pueden formar parte del dato.

Ejemplo

```
LET X1$="TO BE OR NOT TO BE"
LET K2=6.7897
LET Q2$="DIJO HAMLET"
LET R$="EINSTEIN DIJO, 'E = M*C^2"
```

1.5

Ejercicios

1. ¿Cuáles de estas instrucciones LET son incorrectas? Explique por qué.

   a) LET Q$="45*56" *b*) LET E$=S

   c) LET 4X=45 *d*) LET T$="THE BEATLES"

e) LET V2=34

f) LET T="WEE"

g) LET X=Y

h) LET M$=Y

i) LET F=K$

2. Complete el diagrama de cajas siguiente:

	T	R	R$	Q$	T$
LET T=45					
LET R=22.22					
LET T$="MICHELLE"					
LET Q$="ESCRIBIERON"					
LET R$="THE BEATLES"					

3. Complete el siguiente diagrama de cajas:

	A	B	C
LET A=3			
LET B=5			
LET C=A			
LET A=B			
LET B=C			

1.6

Intercambio del contenido de las variables

En el último ejercicio (1.5.3), los números 3 y 5 empiezan fuera de las variables A y B, pero terminan cambiando de lugar. Decimos, entonces, que los contenidos de A y B han sido intercambiados o conmutados.

Demos valores a las variables X e Y:

	X	Y	Z
	0	0	0
LET X=2	2	0	0
LET Y=7	2	7	0

Nuestra intención es intercambiar las posiciones del 2 y el 7. Para hacer esto requerimos una tercera variable para el almacenamiento temporal. Usemos Z.

Las siguientes órdenes LET intercambiarán los valores de X e Y. *Nótese que los números no son usados en el lado derecho del signo "=".*

	X	Y	Z
LET Z=X	2	7	2
LET X=Y	7	7	2
LET Y=Z	7	2	2

¡Hecho! He aquí un ejemplo en el que primero se asignan valores a K1$ y Q$, y después se intercambian.

Ejemplo

	K1$	Q$	P$
LET K1$="HOLA"	HOLA		
LET Q$="PUNT. BAJA"	HOLA	PUNT. BAJA	
LET P$=Q$	HOLA	PUNT. BAJA	PUNT. BAJA
LET Q$=K1$	HOLA	HOLA	PUNT. BAJA
LET K1$=P$	PUNT. BAJA	HOLA	PUNT. BAJA

1.6

Ejercicios

1. Complete este diagrama de cajas e intercambie los contenidos de F y G.

	F	G	X
	0	0	0
LET F=3	3	0	0
LET G=8	3	8	0
LET X=F			
LET F=G			
LET G=X			

2. Complete esta secuencia de instrucciones LET y el diagrama de cajas adjunto para intercambiar los contenidos de T y P. No mire al problema 1 a menos que le sea imprescindible.

	T	P	Z
	0	0	0
LET T=7	7	0	0
LET P=9	7	9	0

3. Complete el diagrama de cajas siguiente:

	X$	Y$	Z1$	K$	P$
LET X$="DULCE"					
LET Y$="BUENA"					
LET Z1$="PRINCIPE"					
LET K$="NOCHE"					
LET P$=X$					
LET X$=Y$					
LET Y$=K$					
LET K$=Z1$					
LET Z1$=P$					
LET P$="!"					

En el ejercicio 1.6.3 anterior usted intercambió los contenidos de más de dos variables. He aquí un ejemplo en el que se hace esto otra vez.

Ejemplo Aquí queremos "rotar" los valores de A, B y C una caja hacia la izquierda. Utilizaremos, como siempre, una variable de almacenamiento temporal.

	A	B	C	R
	0	0	0	0
LET A=3	3	0	0	0
LET B=1	3	1	0	0
LET C=2	3	1	2	0

¡AHORA ROTAMOS!

LET R=A	3	1	2	3
LET A=B	1	1	2	3
LET B=C	1	2	2	3
LET C=R	1	2	3	3

Ejemplo Veamos si podemos escribir algo de Shakespeare. Comencemos con este diagrama de cajas:

	A$	Q$	R$	V$	G1$	A2$
LET A$="OR"	OR					
LET V$="TO"	OR			TO		
LET A2$="NOT"	OR			TO		NOT
LET G1$="BE"	OR			TO	BE	NOT

Escribamos ahora una secuencia de órdenes LET que finalice con una fila de cajas en la que se pueda leer "TO BE OR NOT TO BE". Hagámoslo copiando e intercambiando los contenidos de las variables. Inténtelo usted antes sin mirar la solución.

He aquí una solución:

	A$	Q$	R$	V$	G1$	A2$
LET R$=A$	OR		OR	TO	BE	NOT
LET Q$=G1$	OR	BE	OR	TO	BE	NOT
LET A$=V$	TO	BE	OR	TO	BE	NOT
LET V$=A2$	TO	BE	OR	NOT	BE	NOT
LET A2$=G1$	TO	BE	OR	NOT	BE	BE
LET G1$=A$	TO	BE	OR	NOT	TO	BE

1.6

Ejercicios (cont.)

4. Obsérvese que este diagrama de cajas "rota" los contenidos de A, B y C hacia la derecha. Pero el diagrama está incompleto. Rellene los valores que faltan y las órdenes apropiadas.

A	B	C	D
0	0	0	0
5	0	0	0

5	4	0	0
5	4	3	0
3	5	4	3

5. Dadas estas tres instrucciones LET:

   LET R$ = "MARY"
   LET Q$ = "JUAN"
   LET T$ = "AMA A"

   use una cuarta variable y una secuencia de órdenes LET, de tal manera que la última línea del diagrama de cajas quede así:

R$	Q$	T$
JUAN	AMA A	MARY

1.7

La instrucción PRINT

La instrucción PRINT se usa para escribir información en el TRC (Tubo de Rayos Catódicos, lo cual es una divertida manera de llamar a la pantalla de video).

Ejemplo

	TRC
PRINT "HOLA"	(HOLA)
PRINT 8	(8)
PRINT "6+2"	(6+2)

El símbolo ◯ es nuestra manera de representar el dato impreso en la pantalla.

Una importante función de la orden PRINT es exhibir en la pantalla los contenidos de las variables.

Ejemplo

	X	K$	TRC
	0		
LET X = 2	2		
PRINT X	2		(2)

LET K$ = "BEATLES"	2	BEATLES	
PRINT K$	2	BEATLES	(BEATLES)
PRINT "GOT IT!"	2	BEATLES	(GOT IT!)

¡Fíjese que PRINT "GOT IT!" afecta a la pantalla pero no a las variables!

NORMAS PARA LA INSTRUCCION PRINT

1. La instrucción PRINT puede imprimir datos numéricos o tipo cadena directamente sobre la pantalla, sin almacenarlos primero en variables. Todos los datos tipo cadena deben ir entre comillas, pero éstas no pueden formar parte del dato.
2. La instrucción PRINT puede imprimir los contenidos de las variables numéricas y tipo cadena sobre la pantalla.
3. La instrucción PRINT nunca cambia los contenidos de una variable, es decir, no es destructiva para las variables.

Ejemplo

	M$	S	TRC	
		0		
LET M$ = "JUEGO"	JUEGO	0		
LET S = 556	JUEGO	556		
PRINT "JUEGO"	JUEGO	556	(JUEGO)	Norma 1
PRINT M$	JUEGO	556	(JUEGO)	Norma 2
PRINT 556	JUEGO	556	(556)	Norma 1
PRINT S	JUEGO	556	(556)	Norma 2

1.7

Ejercicios

1. Rellene el diagrama de cajas partiendo de las instrucciones LET y PRINT siguientes:

	V	S	R$	TRC
	0	0		
PRINT V				( )
LET V = 17				
PRINT V				( )
LET S = 42				

PRINT S

LET V=S

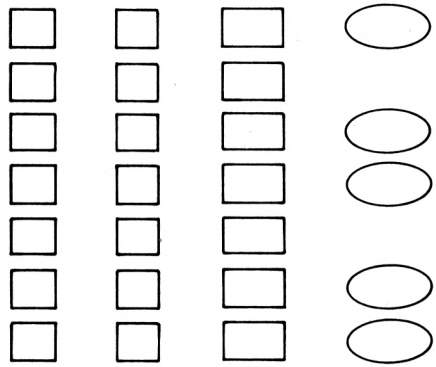
PRINT "V YA ESTA"

PRINT V

LET R$="ESTO ES DIVERTIDO"

PRINT R$

PRINT "FIN"



1.8

Haciendo aritmética

Las operaciones aritméticas usuales se pueden realizar en BASIC. A continuación se indican los símbolos utilizados para realizar operaciones con el ordenador:

^	Exponenciación	−	Resta
*	Multiplicación	+	Suma
/	División		

Exponenciación quiere decir «elevar a una potencia». Esto es, 5^3 significa 5 elevado a la tercera potencia, o 5*5*5. Por tanto, 5^3 es igual a 125.

Las operaciones aritméticas son utilizadas en muchas instrucciones BASIC. Normalmente, las operaciones aritméticas se hacen en una instrucción LET.

Ejemplo

	X
	0
LET X=2+3	5

El ordenador realiza la operación aritmética 2+3, y entonces asigna la solución a la caja X.

Ejemplo

	X	R	U
	0	0	0
a) LET X=1+1	2	0	0
b) LET R=X*3	2	6	0
c) LET U=R/X	2	6	3

Las líneas *b*) y *c*) en este diagrama de cajas son muy importantes. Muestran que las operaciones aritméticas pueden ser hechas sobre el dato que está dentro de una caja *usando el nombre de la caja en el problema aritmético.*

La línea *b*) multiplica el contenido de X por 3 y asigna la solución a la caja R. La línea *c*) utiliza los números hallados en las cajas R y X para hacer una operación de división. La respuesta 3 es asignada a la caja U.

¡Ahora ponga atención! Está usted a punto de aprender algo muy importante. *Es posible hacer aparecer a una misma variable a ambos lados del signo* "=". Consideremos el siguiente diagrama de cajas:

	X
	0
LET X=5	5
LET X=X+7	12

Un matemático podría sorprenderse ante LET X=X+7, pero para el ordenador esto es completamente lógico. Primero, el valor 5 ya está colocado en X; así pues, el valor de X (5) es sumado a 7, resultando 12. La respuesta es entonces colocada *dentro de la caja* X, la cual destruye el 5 que estaba allí. Recuerde la norma 2 para la instrucción LET. *La asignación siempre ocurre de derecha a izquierda.* Esto le ayudará a comprender por qué una misma variable puede estar en ambos lados.

Ejemplo

	X	Y
	0	0
LET X 4	4	0
LET Y=X^3	4	64
LET X=Y/X	16	64
LET Y=Y−X	16	48

1.8 Ejercicios

1. Complete el siguiente diagrama de cajas:

	X	A	C3
	0	0	0
LET X=4			
LET A=X−2			
LET C3=A*X			
LET X=C3^A			

2. Complete el siguiente diagrama de cajas:

	E	F
	0	0
LET E=3		
LET F=E*E		
LET F=F−E		
LET F=F−E		
LET E=F^E		

¿Qué valor debería ser asignado a X con la instrucción
LET X=8−4+2?

Si el ordenador resta 4 de 8 y luego suma 2, la solución es 6. Pero si suma 4 y 2 y luego resta el resultado de 8, ¡la solución que resulta es 2! ¡Ambiguo, ¿eh?! Esto podría ocurrir si no fuera por un conjunto de normas llamado:

ORDEN DE LAS OPERACIONES ARITMETICAS

El ordenador realiza las operaciones aritméticas en el siguiente orden:

1. Primero hace las operaciones que están entre paréntesis.
2. Los signos positivo (+) y negativo (−) se consideran antes de que se haga cualquier otra operación aritmética. Así, −2^4 es lo mismo que (−2)^4 ó 16.
3. La exponenciación se realiza antes que las demás operaciones.
4. La multiplicación y la división se hacen a la vez, de izquierda a derecha.
5. La suma y la resta se realizan al final, de izquierda a derecha.

Ejemplos

	W
	0
LET W=10−4+5	11
LET W=10−(4+5)	1
LET W=20+3*4	32
LET W=(20+3)*4	92
LET W=4^3−1+8/2	67
LET W=(5^3+25)/15*10	100

(Si su respuesta al último ejemplo es 1, usted multiplicó antes de haber dividido.)

Finalmente, debe decirse que las operaciones aritméticas pueden aparecer también en una orden PRINT. Esto permite operaciones aritméticas directamente, sin tener primero que asignar valores a las variables.

Ejemplo

	TRC
PRINT 2*5+8/4	(12)

1.8

Ejercicios (cont.)

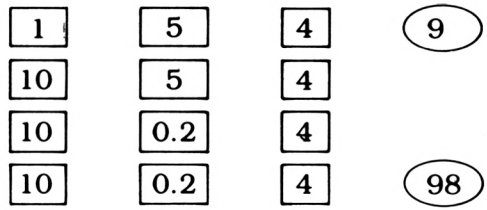
3. Complétese este diagrama de cajas:

	A	TRC
	[0]	
LET A=6	[ ]	
PRINT A*3+2	[ ]	( )
PRINT A/3*2	[ ]	( )
LET A=(A−4)^2	[ ]	
PRINT A−A/4	[ ]	( )
PRINT (A−A)/4	[ ]	( )
LET A=A*A−A+A/A	[ ]	
PRINT "A*A−A+A/A"	[ ]	( )

4. Escriba a la izquierda de cada una de las líneas del diagrama de cajas siguiente una orden LET o PRINT adecuada. Utilice sólo los datos numéricos ya almacenados en las variables.

Puede haber más de una solución en cada una de las líneas del diagrama. Escoja la "más sencilla". La orden LET X=K/K se da como ejemplo:

	X	K	F	TRC
	[0]	[0]	[0]	
LET K=5	[0]	[5]	[0]	
LET X=K/K	[1]	[5]	[0]	
	[1]	[5]	[4]	



Ejemplo La fórmula del capital total obtenido al cabo de un año de inversión bajo un cierto interés simple es

$$A = P*(1 + R)$$

en donde P: Total originalmente invertido, en dólares.

R: Tasa de interés anual.

A: Capital obtenido al año de inversión.

Ejemplo de aplicación El señor Pérez decide invertir 2500 pesetas en una cuenta de ahorro que paga el 15 por 100 de interés en un año. Escriba una secuencia de órdenes LET y PRINT para calcular e imprimir el capital obtenido.

Solución

```
LET P=2500
LET R=.15
LET A=P*(1+R)
PRINT A
```

1.8

Ejercicios (cont.)

5. Realice un diagrama de cajas que explique el ejemplo anterior.
6. La fórmula para el capital total obtenido tras una inversión a interés compuesto, durante N años, es

$$A = P*(1 + R)\hat{}N$$

Escriba una secuencia de órdenes LET y PRINT para calcular el capital total restituido al señor Pérez después de cinco años. Use los valores de R y P del ejemplo anterior.

Ejemplo En matemáticas, el *factorial* de un número natural N (simbolizado por N!) es el producto de todos los números naturales iguales y menores a N (los números naturales son el 1, 2, 3, 4, etc.). Por ejemplo:

```
1! = 1
2! = 2*1 = 2
3! = 3*2*1 = 6
4! = 4*3*2*1 = 24
5! = 5*4*3*2*1 = 120
```

He aquí una forma para calcular e imprimir números factoriales desde 1! a 5!:

```
LET X1=1
LET X2=2*X1
LET X3=3*X2
LET X4=4*X3
LET X5=5*X4
PRINT X1
PRINT X2
PRINT X3
PRINT X4
PRINT X5
```

1.8

Ejercicios (cont.)

7. Realice un diagrama de cajas para el ejemplo anterior.
8. Descubra la secuencia de formación del diagrama de cajas siguiente. El diagrama muestra otra manera de generar números factoriales, utilizando sólo dos variables y el número 1. Complete la secuencia de instrucciones LET y PRINT para generar e imprimir los números hasta el 5!

	X	Y	TRC
	0	0	
LET X=1	1	0	
LET Y=1	1	1	
PRINT X	1	1	1
LET Y=Y+1	1	2	
LET X=X*Y	2	2	
PRINT X	2	2	2
LET Y=Y+1	2	3	
LET X=X*Y	6	3	
PRINT X	6	3	6
.	.	.	.
.	.	.	.
.	.	.	.

Ejemplo La señora Gómez adquirió 250 acciones de IBM a 7625 pesetas cada acción y más tarde vendió las acciones a 8150 pesetas cada una. Escriba la

secuencia de órdenes LET y PRINT para calcular e imprimir el total invertido y su beneficio neto.

Solución

```
LET S=250
LET C1=7625
LET T1=S*C1
PRINT T1
LET C2=8150
LET T2=S*C2
LET P=T2-T1
PRINT P
```

1.8

Ejercicios (cont.)

9. Realice un diagrama de cajas para el ejemplo anterior.
10. Suponga que la señora Gómez recibió dividendos de 450 pesetas por cada acción mientras ella disponía de las acciones. Modifique la anterior secuencia de tal manera que sus dividendos totales sean ordenados dentro de su beneficio.
11. De sus dividendos, ella pagó un 5 por 100 de comisión al comprar las acciones y un 7 por 100 cuando las vendió. Modifique la secuencia del problema 10 para reflejar estos costes en su beneficio neto.

1.9

Proyectos

1. La señora Rincón compró 134 acciones de IBM a 7550 pesetas cada acción. Recibió dividendos de 250 pesetas por cada acción y más tarde las vendió a 8875 pesetas cada una. Escriba una secuencia de órdenes LET y PRINT que calculen e impriman el precio total de compra, el total de dividendos recibidos, el precio total de venta y los beneficios netos recibidos, siendo

   PRECIO TOTAL DE COMPRA = # ACCIONES COMPRADAS *PRECIO DE COMPRA DE CADA ACCION.

   TOTAL DE DIVIDENDOS = # ACCIONES COMPRADAS * DIVIDENDOS RECIBIDOS POR CADA ACCION.

   PRECIO TOTAL DE VENTA = # ACCIONES COMPRADAS * PRECIO DE VENTA DE CADA ACCION.

   BENEFICIOS NETOS = PRECIO TOTAL DE VENTA + TOTAL DE DIVIDENDOS – PRECIO TOTAL DE COMPRA.

2. El señor López compró un automóvil nuevo por 1000000 de pesetas. Pagó 100000 pesetas al contado y aplazó el resto al 1.5 por 100 cada mes con pagos mensuales de 20000 pesetas. A continuación se explica cómo son computados los intereses, la amortización de capital y el balance de la deuda para los primeros dos meses.

   PRECIO DE COMPRA DEL AUTOMOVIL = 1000000
   CANTIDAD APLAZADA = 1000000 − 100000 = 900000
   PAGOS MENSUALES = 20000

   AL FINAL DEL PRIMER MES:

   INTERESES = 900000 * .015 = 13500
   AMORTIZACION DE CAPITAL = 20000 − 13500 = 6500
   BALANCE DE LA DEUDA = 900000 − 6500 = 893500

   AL FINAL DEL SEGUNDO MES:

   INTERESES = 893500 * .015 = 13403
   AMORTIZACION DE CAPITAL = 20000 − 13403 = 6597
   BALANCE DE LA DEUDA = 893500 − 6597 = 886903

   Escriba una secuencia de órdenes LET y PRINT que calcule e imprima el interés, el pago justo y el balance de la deuda para tres meses. Compare sus soluciones con los dos primeros meses de los números anteriores.

3. Guille depositó 1500 pesetas en una cuenta de ahorro que paga el 15 por 100 al año. Decide mantener las 1500 pesetas y todos los intereses devengados en la cuenta hasta que en ésta haya, al menos, 2500 pesetas. El método para calcular el interés compuesto es como sigue:

   AL FINAL DEL PRIMER AÑO:

   INTERES PAGADO A LA CUENTA = 1500 * .15 = 225
   CANTIDAD TOTAL EN LA CUENTA = 1500 + 225 = 1725

   AL FINAL DEL SEGUNDO AÑO:

   INTERES PAGADO A LA CUENTA = 1725 * .15 = 258.75
   CANTIDAD TOTAL EN LA CUENTA = 1725 + 258.75 = 1983.75

   Escriba una secuencia de órdenes LET y PRINT que calcule e imprima:

   a) El interés pagado a la cuenta cada año.
   b) Cantidad total en la cuenta cada año hasta que al menos tenga 2500 pesetas en la cuenta.

4. π es un importante número en matemáticas. Una de las propiedades interesantes de π es que la secuencia de sus decimales no es periódica:

   $$\pi = 3.1415926...$$

La siguiente secuencia de sumas puede aproximar π tan precisamente como se desee:

$4 - 4/3$
$4 - 4/3 + 4/5$
$4 - 4/3 + 4/5 - 4/7$
$4 - 4/3 + 4/5 - 4/7 + 4/9...$

Escriba una secuencia de instrucciones LET y PRINT que dé 3.1 como una aproximación a π. Esto requiere al menos nueve términos. ¡Conseguir 3.14 requiere 119 términos! Precisar el quinto decimal requiere unos 200,000 términos. ¡Como se ve, necesitamos aprender un poco más antes de que podamos abordar esto!

PRACTICA 1.1

La instrucción LET

Introduzca la orden: NEW

Después de haber escrito una orden, pulse la tecla RETURN.

La instrucción LET le da la posibilidad de asignar datos en la memoria del ordenador.

Introduzca la orden: LET X=4

Pulse la tecla RETURN, después de haber escrito esta orden.

Cuando el ordenador ejecuta esta orden, asigna el dato numérico 4 a la caja X. Así pues, el número 4 está ya almacenado en la caja X.

Introduzca la orden: LET 3=X

¿Cuál es la respuesta del ordenador? _____

El ordenador está programado para dar la respuesta ?SINTAX ERROR (error sintáctico) siempre que se intente ejecutar una instrucción con sintaxis errónea.

Se produjo ?SINTAX ERROR porque en una instrucción LET, una variable sólo puede aparecer a la izquierda del signo igual ("=").

Introduzca la orden: LET X=7

¿Qué hay ahora en la caja X? _____

Usted debe haber respondido 7. La instrucción LET asigna y almacena los datos dentro de la memoria del ordenador. Cuando éste ejecutó la orden

LET X=7, el número 4 fue reemplazado por el número 7 en la caja X. A esto se le llama «lectura destructiva», pues el número 4 fue borrado cuando el valor 7 se asignó a X. Solamente puede ser almacenado un valor en una variable al mismo tiempo.

Introduzca la orden: LET Y=X

¿Qué contiene la caja Y? _____

Usted debe haber respondido 7. Cuando el ordenador ejecutó la orden LET Y=X, primero hizo una copia electrónica del número almacenado en la caja X. Esto se llama «lectura no destructiva», pues el número 7 no es borrado de la caja X. A continuación, el ordenador asignó esta copia electrónica, el número 7, a la caja Y.

¿Qué hay contenido en la caja X? _____

Si usted dijo 7, entonces ¡ha comprendido! La orden LET Y=X simplemente copia el valor de X dentro de Y, pero no destruye el contenido de X.

Recuerde esto siempre: La instrucción LET asigna el valor de la derecha del signo igual a la variable que está a su izquierda.

PRACTICA 1.2

La instrucción PRINT

Introduzca la orden: NEW

Después de haber introducido cualquier orden, pulse la tecla RETURN.

La instrucción PRINT crea una señal de salida que usualmente va a una impresora o al TRC (tubo de rayos catódicos), y hace posible ver los datos del ordenador. La instrucción PRINT puede utilizarse para visualizar en pantalla los datos numéricos.

Introduzca la orden: PRINT 4

No se olvide de pulsar la recla RETURN después de haber escrito esta orden.

¿Cuál es la respuesta del ordenador? _____

Usted debe ver el número 4 en su pantalla debajo de la orden PRINT 4.

Los datos tipo cadena también pueden ser visualizados usando la instrucción PRINT.

Introduzca la orden: PRINT "HOLA"

El ordenador escribe HOLA sin las comillas. Una cadena es una colección de caracteres. Para visualizar una cadena, póngala entre comillas a continuación de la instrucción PRINT. El signo de las comillas no puede ser encerrado entre comillas.

¿Qué orden debe ser escrita en el ordenador para que éste visualice la cadena PROGRAMACION BASIC? _____

_____

Usted debe haber escrito: PRINT "PROGRAMACION BASIC".

Volvamos a los datos numéricos.

Introduzca la orden: PRINT 123456789

¿Cuál es la salida del ordenador? _____

La respuesta 123456789 no debe ser una sorpresa.

Introduzca la orden: PRINT 1234567890

¿Cuál es la respuesta del ordenador? _____

La respuesta 1.23456789E+09 está escrita en notación científica. Esto significa 1.23456789 veces 10 elevado a la novena potencia, lo cual es, en efecto, 1234567890. El lenguaje BASIC emplea la notación científica para expresar números muy grandes o muy pequeños.

Introduzca la orden: PRINT .01

¿Cuál es la respuesta del ordenador? _____

Introduzca la orden: PRINT .009

¿Cuál es la salida del ordenador? _____

En el segundo caso la salida está en notación científica. Los números mayores de 999999999 o menores que .01 son visualizados en notación científica con no más de nueve dígitos significativos.

Introduzca las órdenes: LET X=7
PRINT X

¿Cuál es la respuesta del ordenador? _____

Usted debe ver el número 7 escrito después de la orden PRINT X. Cuando

el ordenador ejecuta la orden PRINT X toma una copia del contenido de la caja X y visualiza esta copia en el TRC (la pantalla). Debe quedar claro que la orden PRINT X no cambia o destruye el contenido de X (lectura no destructiva).

PRACTICA 1.3

Variables numéricas y de cadena

Escriba la orden: NEW

No olvide pulsar la tecla RETURN después de haber escrito una orden.

Una variable es el nombre de una «caja» o área de almacenamiento en la memoria del ordenador. Existen dos tipos de variables usadas en BASIC. Son las variables *numéricas* y *tipo cadena*. Ambos tipos de variables tienen nombres que son combinaciones de letras y dígitos, pero deben comenzar con una letra. Además, los nombres de las variables tipo cadena deben terminar con el signo del dólar, $. Por ejemplo, X y TAX82 son variables numéricas y almacenan datos numéricos. Ejemplos de variables tipo cadena son X$ y TAX82$, los cuales pueden usarse para almacenar datos tipo cadena.

El valor «presupuesto» por omisión para una variable numérica es cero (0). Esto significa que si a una variable numérica no se le ha asignado un valor, ésta valdrá cero.

Escriba la orden: PRINT X

¿Cuál es la salida del ordenador? _____

Ya que ningún valor le ha sido asignado a X, el ordenador le asignó el valor cero, por omisión. El valor presupuesto por omisión para una variable de cadena es una cadena sin caracteres, llamada «cadena nula».

Escriba la orden: PRINT X$

¿Cuál es la respuesta del ordenador? _____

Usted está viendo una cadena nula. Esta es la razón por la cual usted no ve ninguna salida en pantalla.

Una cadena es un conjunto de caracteres. Para asignar una cadena a una variable tipo cadena con una instrucción LET, usted debe encerrar entre comillas a la cadena.

Introduzca las órdenes:
```
LET X$="HOLA"
PRINT X$
```

¿Cuál es la respuesta del ordenador? _____

Fíjese que las comillas no se han impreso. Esto es debido a que las comillas son usadas por el ordenador para identificar el principio y el final de una cadena, por lo cual no pueden utilizarse como caracteres dentro de dicha cadena.

Escriba la orden: LET X="5"

¿Cuál es la respuesta del ordenador? _____

X es una variable numérica y sólo aceptará datos numéricos. El "5" es una cadena. El ordenador no almacenará una cadena en una variable numérica. Aparecerá un error del tipo TYPE MISMATCH ERROR, como usted ha visto.

Escriba las órdenes: LET X=5
PRINT X

¿Cuál es la respuesta del ordenador? _____

Ya que el 5 no está entre comillas, el ordenador ve al 5 como dato numérico.

Como se mencionó al principio de esta práctica, los nombres de las variables pueden tener varios caracteres de longitud (de hecho, hasta 238). La mayor parte de las combinaciones de letras y dígitos pueden representar una variable. Sin embargo, hay palabras reservadas que usted no puede emplear dentro de los nombres de las variables.

Introduzca la orden: LET TOTAL=345

¿Cuál es la respuesta del ordenador? _____

La palabra TO es una palabra reservada. Usted no puede escribir TO nunca en un nombre de variable. De ahí la respuesta ?SINTAX ERROR. Existe una lista completa de palabras reservadas en el apéndice B del texto.

Introduzca las órdenes: LET PAGUE=345
PRINT PAGUE

¿Qué valor es almacenado en la variable PAGUE? _____

Introduzca la orden: PRINT PAGO

¿Cuál es la salida del ordenador? _____

¿Sorprendido? Ambas variables PAGUE y PAGO tienen el mismo valor, aunque a la variable PAGO no le haya asignado el valor 345. El Applesoft

BASIC reconoce sólo los dos primeros caracteres del nombre de las variables. Ya que tanto PAGUE como PAGO tienen iguales los dos primeros caracteres, el ordenador ve a ambas como la misma variable.

Escriba la orden: PRINT PA

Explique por qué la respuesta del ordenador es 345 _____

Introduzca las órdenes: LET NOMBRE$="JOSE FRIAS"
PRINT NO$

¿Cuál es la respuesta del ordenador? _____

Introduzca la orden: PRINT NO

¿Cuál es la respuesta del ordenador? _____

Explique por qué el ordenador imprimió un 0 en vez de JOSE FRIAS _____

_____

_____

_____

PRACTICA 1.4

Intercambio

Introduzca la orden: NEW

Muchas aplicaciones del ordenador requieren que los contenidos de dos variables diferentes sean intercambiados. Para ordenar una lista de números, por ejemplo, puede ser necesario pasar lo que esté almacenado en X a Y y lo que esté almacenado en Y a X. Esto se llama *intercambio* de los datos de dos variables.

Introduzca las órdenes: LET X=5
LET Y=2

¿Qué valor está almacenado en la variable X? _____

¿Qué valor está almacenado en la variable Y? _____

El número 5 está almacenado en la variable X y el número 2 es almacenado en la variable Y. Supongamos que no sabemos qué datos están almacenados en X e Y. Intercambiar los datos de X e Y, por tanto, no puede

depender de un conocimiento previo de cuáles son los datos. Para hacer el intercambio requerimos una tercera variable que llamaremos P.

Introduzca las órdenes:
```
LET P=X
LET X=Y
LET Y=P
```

La primera orden, LET P=X, asigna a P una copia del dato que está dentro de X. La segunda orden, LET X=Y, destruye el dato que está dentro de X para reemplazarlo por una copia del dato que está dentro de Y. La tercera orden, LET Y=P, asigna a Y el dato que está dentro de P. Pero recuerde, el dato almacenado en P es el dato original de X. Así, Y se convierte en X, y X se convierte en Y. ¡Un intercambio perfecto!

Suponga que un estudiante escribe la siguiente secuencia de órdenes para intercambiar los contenidos de X e Y:

```
LET X=8
LET Y=3
LET X=Y
LET Y=X
```

Explique por qué esta particular secuencia de órdenes no conmuta los contenidos de X e Y.

_____

La tercera orden, LET X=Y, reemplazará el contenido de X por una copia del contenido de Y. En este punto, tanto X como Y contienen un 3. ¡Piense acerca de esto! Ya no hay ningún 8 que pueda recuperarse. El cuarto paso no hace más que reasignar 3 a Y. Así pues, tanto X como Y tendrán el dato 3.

Introduzca las órdenes:
```
LET X$="HOLA"
LET X$="ALLI"
```

¿Qué secuencia de órdenes se necesita introducir en el ordenador para intercambiar los contenidos de X$ e Y$ usando sólo variables a ambos lados del signo igual? _____

_____

Introduzca las órdenes que usted ha escrito. Después imprima X$ *e* Y$ para ver si logró hacerlas intercambiar. He aquí dos posibles maneras de hacerlo:

1)
```
LET Z$=X$
LET X$=Y$
LET Y$=Z$
```

2)
```
LET Z$=Y$
LET Y$=X$
LET X$=Z$
```

No hay nada de particular en el uso de Z$ como tercera variable. Sería válida cualquier variable de cadena diferente de X$ e Y$.

PRACTICA 1.5

Las operaciones aritméticas

Introduzca la orden: NEW

Hay cinco operaciones aritméticas en BASIC. Son la suma (+), la resta (−), la multiplicación (*), la división (/) y la exponenciación (^).

Introduzca las órdenes:
```
PRINT 2+3
LET C=5*9
PRINT C
```

¿Cuál es la respuesta del ordenador? _____

Como usted ve, por supuesto, son las soluciones a los problemas aritméticos. El BASIC siempre evalúa la expresión aritmética antes de imprimirla o asignarla.

El orden en el cual son realizadas las operaciones de una expresión aritmética sigue estas normas:

1. Se realizan primero las operaciones que están entre paréntesis.
2. La exponenciación se realiza antes que las otras cuatro operaciones aritméticas.
3. Las operaciones multiplicación y división son realizadas antes que las operaciones de suma y resta.

En la mayor parte de los teclados el símbolo de la exponenciación (^) es el SHIFT-N, que consiste en pulsar las teclas SHIFT y N simultáneamente.

Introduzca la orden: PRINT 5+3*4^2

¿Cuál es la respuesta del ordenador? _____

Use las normas de orden de las operaciones para explicar por qué la respuesta es 53 __

_____

_____

_____

La operación se realizó según los siguientes pasos:

Paso 1:	5+3*16	La exponenciación se hace antes que la multiplicación o la división.
Paso 2:	5+48	La multiplicación se hace antes que la suma.
Paso 3:	53	Suma 5 y 48.

Introduzca la orden: PRINT (3+5)^2

¿Cuál es la respuesta del ordenador? _____

Explique cómo el ordenador llega a la solución 64.

_____

_____

La operación se realizó según los siguientes pasos:

Paso 1: 8^2 La operación de dentro del paréntesis se hace primero.

Paso 2: 64 La exponenciación de 8^2. 8^2 significa 8*8.

Introduzca la orden: LET X=2*(3+1)^2/(9−1)+7

¿Cuál es el valor almacenado en la variable X? _____

Introduzca la orden: PRINT X

Explique los siguientes pasos, los cuales muestran cómo el ordenador llegó a 11:

Paso 1: 2*4^2/8+7 _____

Paso 2: 2*16/8+7 _____

Paso 3: 4+7 _____

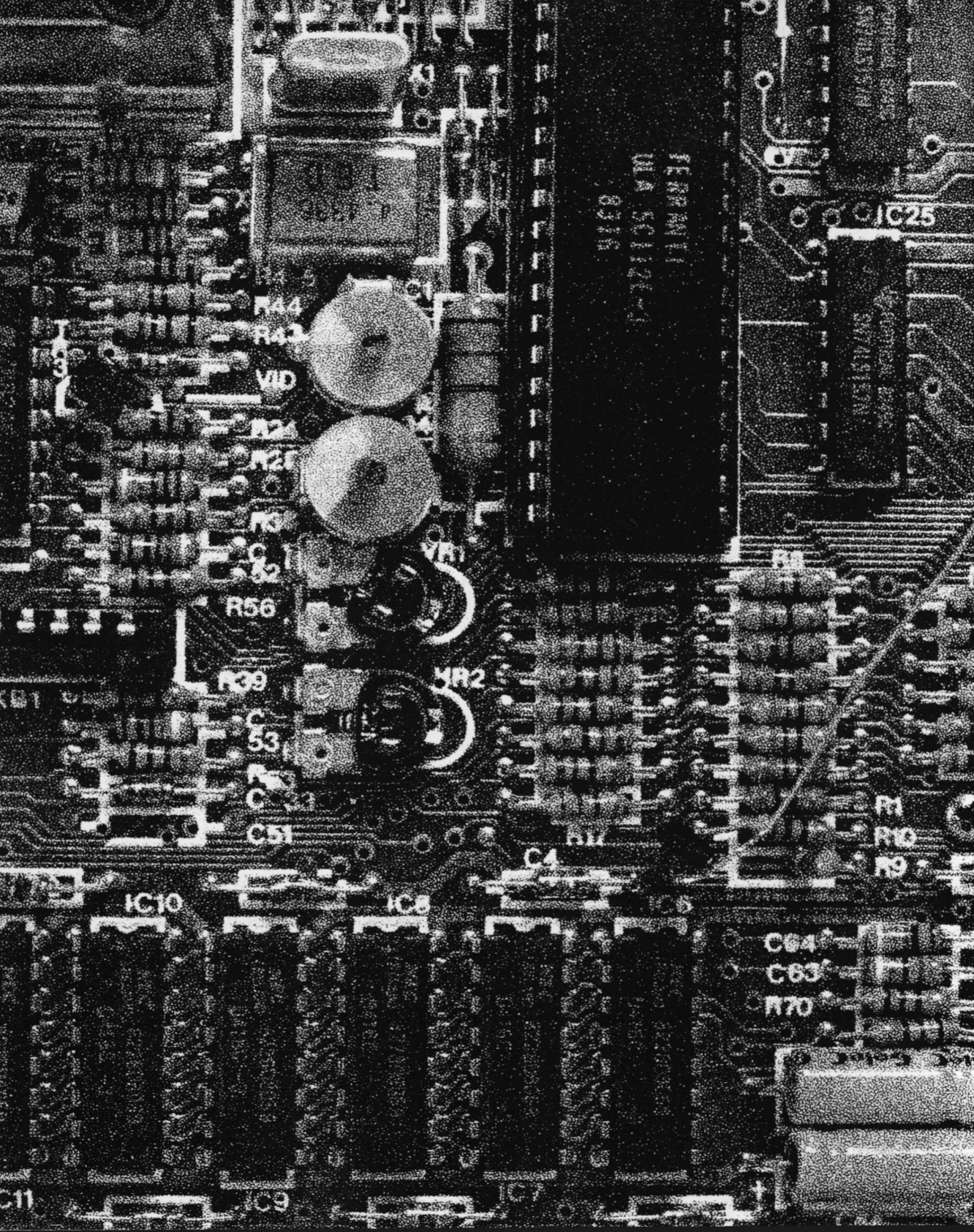
Paso 4: 11 _____

En el paso 1 son efectuadas las operaciones en el interior de los paréntesis. Paso 2, exponenciación de 4^2. Paso 3, el producto y el cociente de 2*16/8. Y paso 4, la suma de 4 y 7.

Introduzca la orden: LET X=2(4−1)

¿Cuál es la respuesta del ordenador? _____

Cada operación aritmética necesita un símbolo. En este caso, el símbolo de la multiplicación (*) ha desaparecido de entre el 2 y (4−1). De ahí el error sintáctico.


IC25
R44
VID
R56
R39
C51
C4
R1
R9
IC10
IC8
C64
C63
R70
IC9
IC7

2
Funciones

2.0
Introducción

Las funciones se pueden considerar como instrucciones incorporadas en ordenador que realizan tareas específicas. Las funciones se dividen en dos categorías: aritméticas y de cadena. Las funciones aritméticas extienden el poder del ordenador para el cálculo más allá de las operaciones simples que estudiamos en el capítulo 1. Pero, mientras muchas personas piensan que el ordenador es sólo una calculadora costosa, su habilidad para procesar datos tipos cadena es, por lo menos, importante.

Todas las funciones están estructuradas de la misma manera (a los programadores les gusta decir que tienen la misma "sintaxis" general). Cada función tiene un nombre, tal como SQR, que sugiere su propósito, en este caso calcular una raíz cuadrada (SQuare Root). El nombre va siempre seguido de paréntesis, como en SQR(X). Lo que va dentro del paréntesis es llamado "argumento" de la función. Una función aritmética opera sobre su argumento y lo transforma en dato numérico. Una función de cadena opera sobre su argumento y lo transforma o bien en datos numéricos o bien en

datos tipo cadena. Las funciones incorporadas siguientes son válidas en el Applesoft BASIC. Aquéllas marcadas con un asterisco son discutidas en el capítulo 12.

Funciones aritméticas	*Funciones de cadena*
ABS(X)	* ASC(X$)
ATN(X)	* CHR$(X)
COS(X)	LEFT$(X$,N)
EXP(X)	LEN(X$)
INT(X)	MID$(X$,J,K)
LOG(X)	RIGHT$(X$,N)
* PDL(X)	STR$(X)
* PEEK(X)	VAL(X$)
RND(X)	
SGN(X)	
SIN(X)	
SQR(X)	
TAN(X)	

21

Las funciones aritméticas

Los argumentos de las funciones aritméticas son siempre valores numéricos, variables numéricas o expresiones aritméticas.

Ejemplo He aquí algunos argumentos de función válidos.

Argumentos válidos	*Expresiones válidas*
3 es un valor numérico	SQR(3)
X es una variable numérica	ABS(X)
4*3+2 es una expresión aritmética	LOG(4*3+2)
X^2+5 es una expresión aritmética	INT(X^2+5)

A) LA FUNCION RAIZ CUADRADA, SQR(X)

Muchas aplicaciones requieren la raíz cuadrada de un número. La expresión SQR(X), cuando se usa en un programa BASIC, realiza la raíz cuadrada del argumento. La función SQR(X) puede usarse igual que un número. Se puede usar en las operaciones aritméticas, en las instrucciones PRINT y LET y en cualquier otro sitio donde estén permitidos datos numéricos.

Ejemplo

	X	Y	TRC
PRINT SQR(49)	0	0	7
LET X=4	4	0	
LET Y=3	4	3	
PRINT SQR(X*X+Y*Y)	4	3	5
LET Y=(SQR(X)+20)/11	4	2	
PRINT SQR(Y)	4	2	1.41421356
PRINT SQR(SQR(X))	4	2	1.41421356

Como muestra la última línea, el argumento de una función puede contener otra función.

2.1

Ejercicios

1. Sea: LET X=3 y LET Y=4. Escriba las instrucciones BASIC apropiadas para calcular las raíces cuadradas de:

   a) X+Y *b)* X*Y *c)* X+SQR((X+4*Y)/3) *d)* X−Y

   Use el ordenador para hallar las respuestas. ¿Puede explicar el resultado que el ordenador consiguió para *d)*?

2. En matemáticas, las ecuaciones de la forma $X^2 - K = 0$, donde K es un número positivo, tienen dos soluciones dadas por

   X1 = SQR(K)
   X2 = −SQR(K)

   Escriba las instrucciones BASIC apropiadas para calcular e imprimir las soluciones de las siguientes ecuaciones:

   $X^2 - 3.4 = 0$
   $X^2 - 7 = 0$
   $X^2 - 150 = 0$

3. La señorita Fernández irá al colegio dentro de dos años. Su educación costará 130000 pesetas, pero por el momento sólo tiene 100000 pesetas. Si quiere poner su dinero en una cuenta de interés compuesto

anual, ¿qué tasa de interés debe elegir? La fórmula de la tasa de interés R, es:

$R = SQR(A/P) - 1$

siendo A el capital total a los dos años y P el capital invertido. Escriba una secuencia de órdenes LET y PRINT que halle el interés que ella necesita para convertir 100000 pesetas en 130000.

4. Las ecuaciones de la forma

$AX^2 + BX + C = 0$

siendo $A > 0$ y $C < 0$, tienen las siguientes dos soluciones:

X1 = (−B + SQR(B^2 − 4*A*C))/(2*A)
X2 = (−B − SQR(B^2 − 4*A*C))/(2*A)

Escriba las instrucciones BASIC apropiadas para calcular e imprimir las soluciones de las siguientes ecuaciones:

$X^2 - 7X - 11 = 0$
$5X^2 - X - 5 = 0$
$14X^2 + 111X - 2 = 0$

B) LA FUNCION VALOR ABSOLUTO, ABS(X)

Podemos convertir un número negativo en otro positivo tomando su "valor absoluto". El valor absoluto de −8 es 8, y el valor absoluto de −23 es 23. El valor absoluto de un número no negativo es el mismo número. Por ejemplo, el valor absoluto de 47 es 47 y el valor absoluto de 0 es 0. La función incorporada en BASIC que determina el valor absoluto de un número X es ABS(X). Como todas las funciones aritméticas, se usa como un número siempre que se necesita.

Ejemplo

	X	Y	TRC
	0	0	
PRINT ABS(−234)	0	0	234
LET X = −5	−5	0	
LET Y = 1	−5	1	
PRINT ABS(X − Y)	−5	1	6
LET X = SQR(ABS(X + Y))	2	1	

Ejemplo La diferencia de altitud entre dos ciudades, D, viene dada por:

```
LET D = ABS(E1-E2)
```

siendo E1 y E2 las altitudes de las dos ciudades. No importa qué altitud es mayor; la diferencia siempre será medida como un número positivo.

Ejemplo La orden

```
LET L = (X + Y + ABS(X - Y))/2
```

siempre asigna a L el mayor de los dos números X e Y.

2.1

Ejercicios (cont.)

5. Escriba una orden LET que asigne a S el menor de los dos números X e Y. Estudie el ejemplo anterior y haga unos ajustes.

C) LA FUNCION PARTE ENTERA, INT(X)

Esta función retorna el mayor entero menor o igual a X.

Ejemplo

X	INT(X)
2.5	2
15.62	15
78	78
0.23	0
−4.2	−5
−35	−35

Parece, con estos ejemplos, que la función INT(X) se comportara de manera diferente con los números positivos que con los números negativos. Pero recuerde que, por ejemplo, −5 es el mayor entero que es menor que −4.2. Para los números positivos la función INT(X) trunca la parte decimal. Por eso mismo, la función INT(X) no redondea los números decimales. Sin embargo, hay un método que usa INT(X) para redondear números decimales largos, tantos dígitos después del punto decimal como se desee (ver la práctica 2.1). A continuación se redondeará el número 9.6872 al número 9.69.

	A
	0
LET A=9.6872	9.6872
LET A=100*A	968.72
LET A=A+.5	969.22
LET A=INT(A)	969
LET A=A/100	9.69

El procedimiento puede ser simplificado a sólo una instrucción:

```
LET A=INT(100*A+.5)/100
```

Al cambiar el número 100 a 1000 redondea A a 3 dígitos decimales, y así sucesivamente.

2.1

Ejercicios (cont.)

6. Escriba una orden LET que redondee los números a un dígito después del punto decimal.
7. Escriba una orden LET que redondee números hasta el dígito de las centenas (esto es, 35874 puede transformarse en 35900).

D) LA FUNCION DEL NUMERO ALEATORIO, RND(X)

Los juegos de azar normalmente requieren un proceso aleatorio tal como el rodar de los dados o el lanzamiento de una moneda. Es fácil conseguir del ordenador números aleatorios utilizando la función RND(X). La función RND(X) generará un número aleatorio mayor o igual a cero, pero menor que uno. El argumento de la función puede ser cualquier número, pero el rango de los números aleatorios siempre será de cero a uno. Por esa razón, ya que es irrelevante, utilizaremos 1 como argumento (un argumento negativo, o el cero, sin embargo, tienen un efecto especial sobre la función RND. Para ejemplos, ver la práctica 2.2).

Ejemplo A continuación se generan tres números aleatorios:

```
                    X
                   [0]
LET X=RND(1)    [.21167342]
LET X=RND(1)    [.89321343]
LET X=RND(1)    [.00553125]
```

En muchas aplicaciones son necesarios números enteros aleatorios. La secuencia de órdenes LET siguiente muestra cómo puede generarse un entero aleatorio entre 1 y 6 (ver práctica 2.2):

	X	
	0	
LET X=RND(1)	.432431	X está entre 0 y 1 (sin incluir 1).
LET X=6*X	2.594586	X está ahora entre 0 y 6. (sin incluir 6).
LET X=INT(X)	2	X es ahora un entero entre 0 y 5.
LET X=X+1	3	X es ahora un entero aleatorio entre 1 y 6.

La secuencia completa puede escribirse con una sola orden LET:

```
LET X=INT(6*RND(1))+1
```

2.1

Ejercicios (cont.)

8. Escriba una secuencia de órdenes BASIC para generar e imprimir cinco enteros aleatorios entre 1 y 20.
9. Escriba una secuencia de órdenes BASIC para generar e imprimir cinco enteros aleatorios entre 6 y 15. Sugerencia: hay 10 números del 6 al 15, use INT(10*RND(1)) y después súmele un número apropiado.

E) OTRAS FUNCIONES ARITMETICAS (OPCIONAL)*

Algunas funciones matemáticas avanzadas están también incorporadas al lenguaje BASIC. Éstas son:

SIN(X)	seno de X radianes.
COS(X)	coseno de X radianes.
TAN(X)	tangente de X radianes.
ATN(X)	arco tangente de X.
LOG(X)	logaritmo neperiano de X.
EXP(X)	e elevado a la potencia de X. siendo e igual a 2.71828183.

* Esta sección puede ser omitida sin pérdida de continuidad. Los estudiantes con buenos conocimientos sobre álgebra y trigonometría pueden encontrar los temas interesantes.

SGN(X)	retorno a +1 si X es positivo, retorno a −1 si X es negativo, retorno a 0 si X es cero.

Ejemplo Un modelo matemático para el crecimiento de la población de un país se da en la fórmula

```
Q = P1 * EXP(K*T1)
```

siendo P1 la población inicial, Q es la población T1 años más tarde y K es la "constante de crecimiento". La constante de crecimiento puede hallarse si se conoce la población del país para otro tiempo. La fórmula de K es

```
K = (LOG(P2) − LOG(P1))/T2
```

siendo P2 la población en otra fecha diferente y T2 el número de años pasados entre las poblaciones P1 y P2.

La población de Uganda en 1969 era de 9.5 millones. En 1977 la población era de 12.3 millones. Calcule la población que se espera para 1985. Las órdenes BASIC siguientes calcularán e imprimirán la población estimada para 1985:

```
LET P1 = 9500000
LET P2 = 12300000
LET T2 = 1977−1969
LET K = (LOG(P2) − LOG(P1))/T2
LET T1 = 1985 − 1969
LET Q = P1 * EXP(K*T1)
PRINT Q
```

2.1 Ejercicios (cont.)

10. La población de Estados Unidos en 1960 era de 179 millones y de 203 millones en 1970. Escriba las instrucciones BASIC apropiadas que predigan la población en el año 2000.

Ejemplo Supongamos que tenemos un triángulo con lados de longitudes A, B y C, y los correspondientes ángulos opuestos A1, B1 y C1. La ley de los cosenos puede usarse para calcular el lado A si los otros dos lados y el ángulo A1 son conocidos:

$$A = \sqrt{B^2 + C^2 - 2BC\,COS(A1)}$$

El siguiente conjunto de instrucciones BASIC calcularán el lado A, siendo B = 24, C = 32 y A1 = 2 (en radianes):

```
LET B = 24
LET = C = 32
LET A1 = 2
LET A = SQR(B^2 + C^2 − 2*B*C*COS(A1))
PRINT A
```

2.1

Ejercicios (cont.)

11. Operando en la ley de los cosenos llegamos a esta interesante fórmula para hallar el coseno del ángulo A1 de un triángulo:

$$\text{coseno de A1} = \frac{B^2 + C^2 - A^2}{2BC}$$

Suponga que tenemos un triángulo con estos datos: A = 1.732, B = 1 y C = 2. Escriba un conjunto de instrucciones BASIC que calculen e impriman el coseno del ángulo A1.

Como sabemos desde los primeros ejemplos, las funciones pueden ser usadas dentro de expresiones aritméticas. *Las funciones siempre tienen prioridad sobre* ^,*,/,+ y −. Puede incluirse esta regla dentro de la tabla del *orden de las operaciones aritméticas* en el apartado 1.8.

Ejemplo En la instrucción

```
LET A = SQR(B^2 + C^2 − 2*C*B*COS(A1))
```

la primera cantidad evaluada es COS(A1), porque tiene preferencia. Después se hace la exponenciación, seguida por la multiplicación, después por la suma, luego por la resta y finalmente se realiza la función SQR. La función SQR se hace al final de este ejemplo porque su argumento ha de ser evaluado primero. Recuerde, el ordenador siempre evalúa primero las expresiones que están entre paréntesis.

2.2

Las funciones con cadenas

Los ordenadores pueden procesar los datos tipo cadena igual que los datos numéricos. Las funciones incorporadas en el BASIC que hacen esto se llaman funciones de cadena.

A) LA FUNCION LONGITUD, LEN(X$)

La longitud de X$ es el número de caracteres de la cadena almacenada en X$. La función LEN(X$) da esta longitud.

Ejemplo

	X$	A	TRC
LET X$ = "HOLA"	HOLA	0	
LET A = LEN(X$)	HOLA	4	
PRINT A	HOLA	4	4
PRINT LEN("AUMENTA LA MARCA")	HOLA	4	16

Es importante fijarse en que la función LEN cuenta *todos* los caracteres de la cadena. En el ejemplo anterior, LEN("AUMENTA LA MARCHA") es igual a 16, ya que los espacios también cuentan como un carácter.

B) LA FUNCION VALOR, VAL(X$)

La orden LET X$ = "54.23" asigna la cadena 54.23 a la variable X$. ¡Pero 54.23 es un número, dirá usted! Dentro del ordenador, la cadena 54.23 se almacena de forma completamente diferente que un número. Algunas veces es necesario para cambiar una cadena por el valor numérico que representa. Esta es la utilidad de la función VAL(X$). La orden

LET A = VAL (X$)

cambia la cadena en dato numérico y lo asigna a A. Los capítulos 9, 10 y 11 tratan importantes aplicaciones de la función VAL(X$).

Ejemplo

	Q$	P
		0
LET Q$ = "65.2"	65.2	0
LET P = VAL(Q$)	65.2	65.2
LET P = P + VAL("22.4")	65.2	87.6

C) LA FUNCION CADENA, STR$(X)

La función STR$(X) es la opuesta de la función VAL(X$). STR$(X) convierte un número en una cadena.

Ejemplo

	T	L$
	0	
LET T=27.82	27.82	
LET L$=STR$(T)	27.82	27.82

De nuevo, veremos más tarde interesantes aplicaciones de la función STR$(X).

D) LAS FUNCIONES SUBCADENA

Acertijo: ¿Cuál es la palabra más larga de la lengua inglesa?

Respuesta: ¡SMILES! (sonrisas), porque entre las dos eses hay una MILE (una milla) de distancia!

El acertijo muestra la noción de "subcadena" de una cadena. MILE es una subcadena de SMILES. Hay tres funciones en el Applesoft que extraen subcadenas. Estas son la función MID$, la función RIGHT$ y la función LEFT$.

1) MID$(X$,J,K)

La función MID$(X$,J,K) hace posible extraer subcadenas de cualquier longitud a partir de una cadena. En la expresión MID$(X$,J,K), la J es la posición de comienzo de dentro de X$, y la K cuenta el número de caracteres que ha de tener la subcadena. Si se omite el número K, como en MID$(X$,J), entonces la subcadena empieza en el J.º carácter de X$ e incluye hasta el final de la cadena.

Ejemplo

	B$	TRC
LET B$="MI CADENA ROTA"	MI CADENA ROTA	
PRINT B$	MI CADENA ROTA	MI CADENA ROTA
PRINT MID$(B$,6,4)	MI CADENA ROTA	DENA
PRINT MID$(B$,8,6)	MI CADENA ROTA	NA ROT
PRINT MID$(B$,14,1)	MI CADENA ROTA	A
PRINT MID$(B$,5)	MI CADENA ROTA	ADENA ROTA
PRINT MID$(B$,LEN(B$)−3)	MI CADENA ROTA	ROTA
LET B$=MID(B$,1,2)	MI	

2) RIGHT$(X$,N)

La función RIGHT$(X$,N) crea una subcadena tomando los N caracteres situados más a la derecha de los que componen X$.

Ejemplo

	B$	TRC
LET B$="BEATLES"	BEATLES	
PRINT RIGHT$(B$,3)	BEATLES	LES
LET B$=RIGHT$(B$,5]	ATLES	
PRINT RIGHT$(B$,1)	ATLES	S

La función RIGHT$(X$,N) puede ser reemplazada por MID$(X$, LEN(X$)−N+1), que produce el mismo efecto. Pero es más conveniente usar RIGHT$.

3) LEFT$(X$,N)

La función LEFT$(X$,N) crea una subcadena tomando los N caracteres situados más a la izquierda de los que componen X$.

Ejemplo

	Y1$	TRC
LET Y1$=LEFT$("CORAZON",4)	CORA	
PRINT LEFT$(Y1$,1)	CORA	C

Nótese que LEFT$(X$,N) es equivalente a MID$(X$,1,N). Así pues, la función LEFT$, como la RIGHT$, son un lujo.

2.2 Ejercicios

1. Sin usar el ordenador, determínese la respuesta a las siguientes órdenes PRINT:

   LET A$="YA VIENE EL CORTEJO"

   a) PRINT LEN(A$)

   b) PRINT MID$(A$,4,3)

   c) PRINT MID$(A$,8,1)

   d) PRINT MID$(A$,15)

   e) PRINT LEN(MID$(A$,15))

f) PRINT RIGHT$(A$,4)

g) PRINT LEFT$(A$,4)

h) PRINT MID$(RIGHT$(A$,8),3,2)

i) PRINT RIGHT$(LEFT$(A$,5),1)

2. ¿Qué errores se han cometido al escribir las siguientes funciones?

a) MID(X$,2,1)

b) STR$("TRES")

c) LEN(43.2)

d) LEFT$(X$,0)

e) MID$(SALCHICHON,3,2)

f) LEN(X$,3,2)

2.3

Proyectos

1. (Sólo para matemáticos.) Muchas de las funciones matemáticas descritas en este capítulo tienen desarrollos de series de Taylor. Una serie de Taylor es un "polinomio infinito" cuyo valor en X coincide con el valor de la función en X. He aquí algunas series de Taylor para algunas de las funciones que hemos visto.

$$\text{EXP}(X) = 1 + X + \frac{X^2}{2} + \frac{X^3}{6} + \frac{X^4}{24} + \cdots$$

$$\text{LOG}(1 + X) = X - \frac{X^2}{2} + \frac{X^3}{3} - \frac{X^4}{4} + \cdots$$

$$\text{COS}(X) = 1 - \frac{X^2}{2} + \frac{X^4}{24} + \cdots$$

$$\text{SIN}(X) = X - \frac{X^3}{6} + \frac{X^5}{120} + \cdots$$

Los primeros términos de una serie de Taylor pueden usarse para aproximar el valor real de una función de X.

a) Use los términos dados en las series de Taylor para calcular un valor aproximado de cada una de las funciones, siendo X igual a 2.

b) Compare las respuestas que obtuvo en el apartado *a*) con las respuestas que obtendría usando las funciones incorporadas del ordenador.

2. La agencia de noticias Ajax acepta anuncios. El coste de un anuncio es de 125 pesetas por línea, teniendo cada línea 55 caracteres (incluidos los espacios). Cualquier porción de línea se considera como una línea completa. El próximo mes aparecerá este anuncio:

   Almacén minorista busca operario, hombre o mujer, para entrar en el excitante campo de las empresas minoristas. Los aspirantes deben estar en posesión del título de graduado escolar. Contactar con el señor Davis en el 555-1423.

   Escriba las instrucciones BASIC apropiadas que calculen el coste total del anuncio.

PRACTICA 2.1

La función INT

Introduzca la orden: NEW

Un entero es un número positivo sin decimales o el opuesto de ese número. Por ejemplo, 5, 0 y −3 son enteros. La instrucción INT(X) genera el mayor entero que es menor o igual que el argumento X; X representa cualquier expresión que represente un número.

Si imaginamos a X como uno de los números de la serie natural de números escrita de izquierda a derecha, entonces INT(X) es igual al primer entero que esté a la izquierda de X, si X no es ya un entero. Si X es un entero, entonces INT(X) es igual a X.

Introduzca la orden: PRINT INT(4.9)

¿Cuál es la respuesta del ordenador? _____

El número 4 es el mayor entero que es menor o igual a 4.9. Nótese que 4 es el primer entero que está a la izquierda de 4.9 en la serie.

Introduzca la orden: PRINT INT(−3.1)

¿Cuál es la respuesta del ordenador? _____

¿Pensó usted que obtendría −3 como respuesta? Recuerde que en la serie natural de números el −4 es el primer entero que está a la izquierda de −3.1. El número −4 es el mayor entero que es menor o igual que −3.1.

Introduzca la orden: PRINT INT(2.9+6.5)

La expresión aritmética 2.9 + 6.5 suma 9.4 y éste es el argumento de la función INT. El ordenador siempre evalúa primero la expresión aritmética que constituye el argumento.

La función INT es de gran aplicación para redondear los números. Por ejemplo, los números 158 y 153 redondeados hasta la decena más próxima son 160 y 150, respectivamente.

La tabla siguiente muestra qué fórmula se utiliza para redondear el número N hasta un lugar dado:

Cuando el número es redondeado hasta este lugar	*La fórmula a usar es:*
Milésimas	INT(1000*N + .5)/1000
Centésimas	INT(100*N + .5)/100
Décimas	INT(10*N + .5)/10
Unidades	INT(N + .5)
Decenas	INT(.1*N + .5)/.1
Centenas	INT(.01*N + .5)/.01
Millar	INT(.001*N + .5)/.001

Escriba la orden PRINT que redondee el número 352.7461 a la centésima más cercana.

Usted debe haber escrito PRINT INT(100*352.7461 + .5)/100.

Introduzca la orden anterior.

La fórmula para redondear cualquier número es

INT(R*N + .5)/R

siendo N el número y R el recíproco del valor relativo al cual se quiere redondear N. Por ejemplo, para redondear el número 352.7461 al lugar de las centésimas, tenemos R = 1/.01 y N = 352.7461.

Introduzca las órdenes:
```
LET N=352.7461
LET R=1/.01
PRINT INT(R*N + .5)/R
```

¿Cuál es el resultado del ordenador? _____

Escriba una secuencia de órdenes que imprima el número 216738 redondeado a la decena de millar más cercana. _____

_____

_____

Usted debe haber escrito:
```
LET N=216738
LET R=1/100000
PRINT INT(R*N+.5)/R
```

Introduzca la secuencia de órdenes anterior.

¿Cuál es la respuesta del ordenador? _____

En resumen, la función INT siempre genera el mayor entero que es menor o igual a su argumento. El ordenador realizará primero en el argumento los cálculos necesarios, antes de generar el entero.

PRACTICA 2.2

La función RND

Introduzca la orden: NEW

La función RND genera un número aleatorio mayor o igual a cero y menor que uno. En Applesoft BASIC se requiere un número para el argumento de la función RND.

Introduzca las órdenes:
```
PRINT RND(1)
PRINT RND(1)
```

¿Cuál es la respuesta del ordenador? _____

_____

Fíjese que los dos números son diferentes. Siempre que el argumento de la función RND sea un número positivo, la función RND generará, cada vez que se ejecute, un nuevo número aleatorio entre cero y uno.

Introduzca la orden: PRINT RND(0)

¿Cuál es la respuesta del ordenador? _____

Deberá ver otra vez el mismo número que anteriormente había generado la función RND. Siempre que el argumento de la función RND sea 0, RND(0) volverá a generar el número aleatorio anterior.

Introduzca las órdenes: PRINT RND(2)
PRINT RND(0)

Explique por qué el ordenador imprimió el mismo número dos veces.

_____

Introduzca las órdenes: PRINT RND(−1)
PRINT RND(−1)
PRINT RND(−1)

¿Cuál es la respuesta del ordenador? _____

¿Son todos los valores iguales? _____

Su respuesta debe ser sí. Si se usa otro número negativo como argumento, la función RND generará un número aleatorio diferente.

Introduzca las órdenes: PRINT RND(−2)
PRINT RND(−2)
PRINT RND(−2)

¿Cuál es la respuesta del ordenador? _____

Escriba las órdenes: PRINT RND(−1)
PRINT RND(−2)

Compare los resultados con las respuestas anteriores.

Escriba las órdenes: PRINT 6*RND(1)
PRINT 6*RND(1)
PRINT 6*RND(1)

¿Cuáles son las respuestas? _____

Usted debería ver números aleatorios mayores o iguales a cero, pero menores que seis.

¿Qué órdenes imprimen números aleatorios mayores o iguales a cero, pero menores que 10?

_____

Usted debe de haber escrito PRINT 10*RND(1).

También podemos generar enteros aleatorios.

Escriba las órdenes:
```
PRINT INT(6*RND(1))
PRINT INT(6*RND(1))
PRINT INT(6*RND(1))
```

¿Cuáles son las respuestas del ordenador? _____

Los números que aparecen son enteros aleatorios del 0 al 5.

Introduzca las órdenes:
```
PRINT INT(6*RND(1))+1
PRINT INT(6*RND(1))+1
PRINT INT(6*RND(1))+1
```

¿Cuáles son las respuestas del ordenador? _____

Los números que está usted viendo son enteros aleatorios del 1 al 6. Esto ilustra cómo la función RND puede usarse para simular los resultados de un dado.

La fórmula para generar un entero aleatorio limitado entre dos enteros incluyendo los límites superior e inferior es

INT(D*RND(1))+L

siendo D la diferencia entre los límites superior e inferior más 1, y L el límite inferior.

Por ejemplo, para generar un entero aleatorio situado entre 3 y 7 tenemos

D=5 (7−3+1=5) y L=3

Introduzca la orden: PRINT INT(5*RND(1))+3

¿Cuál es la respuesta del ordenador? _____

Escriba una orden que imprima un número entero aleatorio situado entre 4 y 9.

Ustedes deben haber escrito: PRINT INT(6*RND(1))−4.

En resumen, la función RND generará números aleatorios diferentes entre 0 y 1 cuando el argumento sea un número positivo. La función RND generará el último número aleatorio generado si el argumento es cero. Y si el argumento de la función RND es negativo, ésta generará el mismo número aleatorio.

PRACTICA 2.3

La función LEN

Introduzca la orden: NEW

La función LEN genera un número igual al número de caracteres de una cadena. El argumento de la función LEN puede ser una cadena encerrada entre comillas, una variable tipo cadena o una expresión de cadenas.

Introduzca la orden: PRINT LEN("HOLA")

¿Cuál es la respuesta del ordenador? _____

Ya que la cadena HOLA tiene cuatro caracteres, la respuesta es 4.

Introduzca las órdenes: LETX$="HOLA"
PRINT LEN(X$)

¿Cuál es la respuesta del ordenador? _____

Fíjese que el argumento de la función LEN es una variable tipo cadena. La respuesta es 4, que es el número de caracteres almacenados en la variable tipo cadena X$.

Introduzca la orden: PRINT LEN("JOSE FRIAS")

¿Cuál es la respuesta del ordenador? _____

Aunque la cadena JOSE FRIAS tiene sólo nueve letras, la respuesta es 10. Esto se debe al carácter en blanco que hay entre las palabras JOSE y FRIAS.

Introduzca las órdenes: LETB$=" " *(tres espacios entre las comillas)*
PRINT LEN(B$)

¿Cuál es la respuesta del ordenador? _____

Introduzca las órdenes: LET C$=“” *(sin espacios)*
PRINT LEN(C$)

¿Cuál es la respuesta del ordenador? _____

Mientras B$ contiene realmente tres espacios en blanco, C$ no tiene ningún caracter. A esta cadena se la llama “cadena nula”.

¿Contiene B$ *una cadena nula?* _____

La respuesta debe ser NO. B$ contiene una cadena de longitud tres. Sin embargo, la cadena nula tiene una longitud cero.

PRACTICA 2.4

La función VAL

Introduzca la orden: NEW

Hay dos tipos de datos usados en BASIC, numéricos y de cadena. La función VAL es utilizada para convertir datos tipo cadena en datos numéricos.

Introduzca las órdenes: LET X$=“123”
LET A=VAL(X$)
PRINT A

¿Cuál es la respuesta del ordenador? _____

Obsérvese que 123 es una cadena almacenada en X$. Cuando el ordenador ejecuta la orden LET A=VAL(X$), la función VAL transformó la cadena 123 en dato numérico y lo asignó a la variable numérica A.

Introduzca las órdenes: LET X$=“4.5 MAS 6.7”
LET A=VAL(X$)
PRINT A

¿Cuál es la respuesta del ordenador? _____

La función VAL intenta interpretar este argumento como un número. Comienza por el principio de la cadena, y cuando encuentra un carácter que no tiene sentido como número, se para. Los espacios, sin embargo, simplemente son ignorados. En el caso de “4.5 MAS 6.7” da como resultado 4.5, pues paró cuando encontró la letra M.

Introduzca las órdenes:
```
LET X$ = "CUATRO"
LET A = VAL(X$)
PRINT A
```

¿Cuál es la respuesta del ordenador?

El primer carácter de la cadena CUATRO es C, el cual no se puede interpretar como un número. Así pues, la función VAL se para ahí y da el valor cero.

Introduzca las órdenes:
```
LET X$ = "12 45"
LET A = VAL(X$)
PRINT A
```

¿Cuál es la respuesta del ordenador?

Recuerde, la función VAL examinará la cadena desde el principio, ignorando los espacios, y se parará sólo cuando encuentre un carácter que pueda formar parte de un número. La respuesta es 1245, ya que el espacio es ignorado.

PRACTICA 2.5

La función STR$

Introduzca la orden: NEW

Hay dos tipos de datos empleados en BASIC, numéricos y de cadena. La función STR$ se usa para convertir datos numéricos en datos cadena.

Introduzca las órdenes:
```
LET X = 123
LET A$ = STR$(X)
PRINT A$
```

¿Cuál es la respuesta del ordenador?

Primero, el dato numérico 123 es almacenado en X. Cuando el ordenador ejecutó la orden LET A$ = STR$(X), la función STR$ transformó el número 123 almacenado en X en una cadena y la asignó a la variable A$.

Introduzca las órdenes:
```
LET X = 4
LET A$ = STR$(X + 5)
PRINT A$
```

¿Cuál es la respuesta del ordenador?

El ordenador siempre evaluará primero la expresión que constituye el argumento. Así, el número 9 fue transformado en una cadena y asignado a A$.

PRACTICA 2.6

La función MID$

Introduzca la orden: NEW

Una subcadena es una parte de una cadena. Para extraer una subcadena de una cadena se utiliza la función MID$.

Introduzca la orden: PRINT MID$("ORDENADOR",4,3)

¿Cuál es la respuesta del ordenador? _____

El argumento de la función MID$ consta de tres partes, separadas por comas. La primera parte puede ser una cadena encerrada entre comillas, una variable tipo cadena o una expresión formada por cadenas.

En la orden anterior, ¿cuál es la cadena de la cual se extraerá la subcadena?

_____

Su respuesta debe ser ORDENADOR. La segunda parte del argumento debe ser una expresión que evalúe un número entero positivo. Este entero da la posición de un carácter de la cadena, que será el primero de la subcadena.

En la orden anterior, ¿qué posición, en ORDENADOR, ocupa el carácter que será el primero de la subcadena? _____

Su respuesta debe ser 4. Esto significa que el cuarto carácter en ORDENADOR es el primer carácter de la subcadena que será extraída. La tercera parte del argumento debe ser una expresión que evalúe un número entero positivo o un cero. Este entero da el número de caracteres que tendrá la subcadena.

En la orden anterior, ¿cuántos caracteres tendrá la subcadena? _____

Su respuesta debe ser 3. La subcadena ENA comienza en el cuarto carácter de la cadena ORDENADOR y tiene tres caracteres de longitud.

Introduzca las órdenes:
```
LET X$="PROGRAMACION BASIC"
PRINT MID$(X$,14,4)
```

¿Cuál es la respuesta del ordenador? _____

El decimocuarto carácter de la cadena almacenada en X$ es la letra B. Recuerde, un espacio se cuenta como un carácter. Tomando cuatro caracteres comenzando con la B, tendremos BASI, lo cual estará ya escrito en la pantalla.

Hay otra forma de la función MID$.

Introduzca la orden: PRINT MID$(X$,10)

¿Cuál es la respuesta del ordenador? _____

El ordenador muestra ION BASIC. En esta forma especial de función MID$ hay sólo un número en el argumento. Este indica en qué carácter debe comenzar la subcadena. La función entonces nos da el resto de la cadena desde esa posición hasta el final.

PRACTICA 2.7

La función RIGHT$

Introduzca la orden: NEW

La función RIGHT$ es un método fácil para extraer los caracteres que están más a la derecha de una cadena.

Introduzca la orden: PRINT RIGHT$ ("ORDENADOR",3)

¿Cuál es la respuesta del ordenador? _____

El argumento de la función RIGHT$ consta de dos partes separadas por una coma. La primera parte puede ser una cadena encerrada entre comillas, una variable tipo cadena o una expresión formada por cadenas.

En la orden anterior, ¿cuál es la cadena de la que se extrae la subcadena?

_____

Su respuesta debería ser ORDENADOR. La segunda parte del argumento debe ser una expresión que evalúe un número entero positivo. Este nos da el número de caracteres que tendrá la subcadena.

En la orden anterior, ¿cuántos caracteres tendrá la subcadena?

_____

Su respuesta debe ser 3. La subcadena impresa es DOR. Los tres caracteres de la subcadena son los tres últimos caracteres de ORDENADOR.

Introduzca la orden: LET X$="PROGRAMACION BASIC"
PRINT RIGHT$(X$,8)

¿Cuál es la respuesta del ordenador? _____

El octavo carácter de la derecha de la cadena PROGRAMACION BASIC es la letra O. Así, la respuesta es ON BASIC. Los últimos ocho caracteres de la cadena PROGRAMACION BASIC forman la subcadena ON BASIO

PRACTICA 2.8

La función LEFT$

Introduzca la orden: NEW

La función LEFT$ nos da un método fácil para extraer los caracteres que estén situados más a la izquierda dentro de una cadena.

Introduzca la orden: PRINT LEFT$("CALCULAR",3)

¿Cuál es la respuesta del ordenador? _____

El argumento de la función LEFT$ consta de dos partes separadas por una coma. La primera parte puede ser indistintamente una cadena encerrada entre comillas, una variable tipo cadena o una expresión formada por cadenas.

En la orden anterior, ¿cuál es la cadena de la que se extraerá la subcadena?

_____

Su respuesta debe ser CALCULAR.

La segunda parte del argumento debe ser una expresión que evalúe un número entero positivo. Éste da el número de caracteres que tendrá la subcadena.

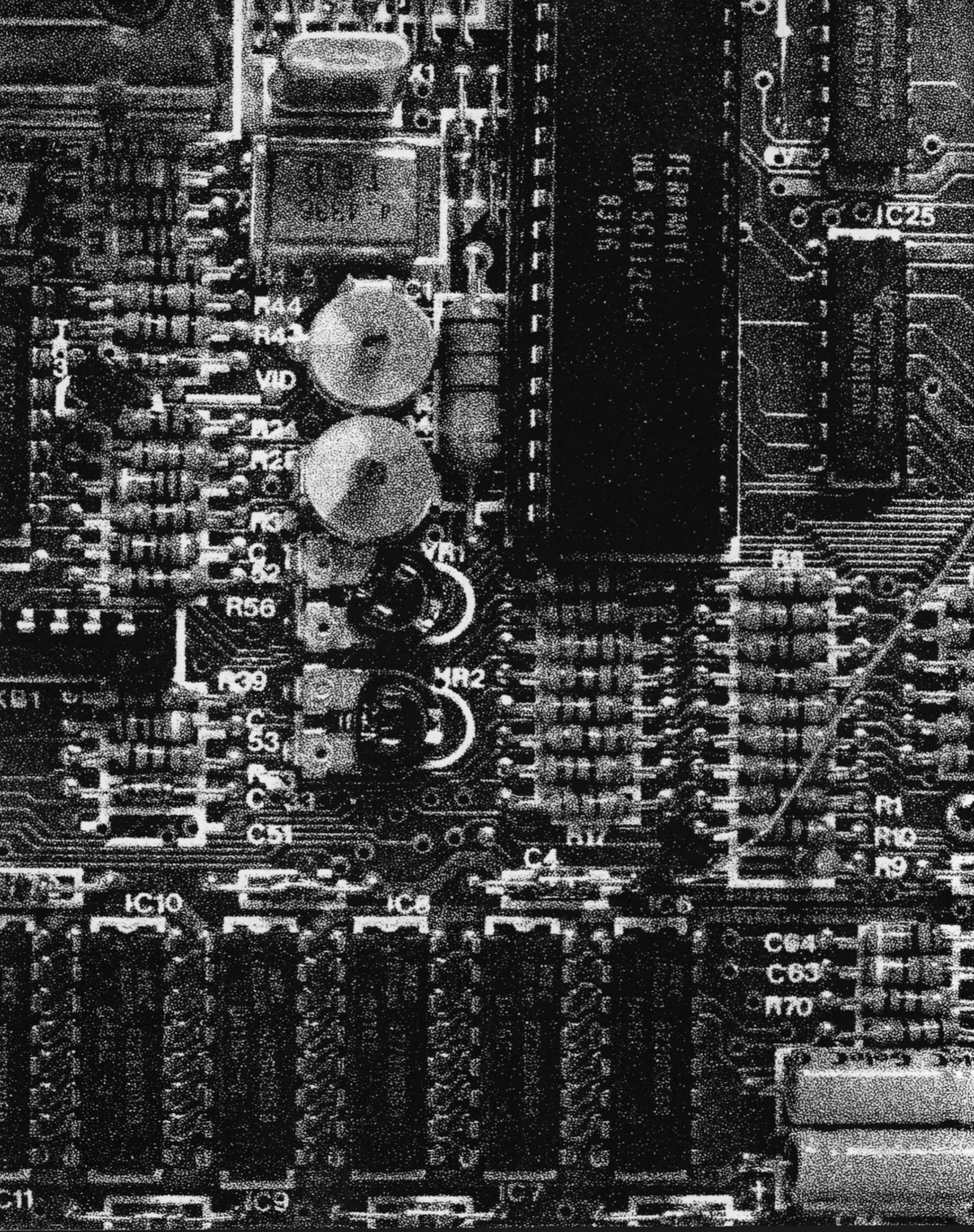
En la orden anterior, ¿cuántos caracteres tendrá la subcadena? _____

Su respuesta debe ser 3. Como ya está viendo en la pantalla, la respuesta para la orden anterior es CAL. Los tres caracteres de esta subcadena son los tres primeros caracteres de CALCULAR.

Introduzca la orden: LET X$="PROGRAMACION BASIC"
PRINT LEFT$(X$,5)

¿Cuál es la respuesta del ordenador? _____

Los cinco primeros caracteres de la cadena PROGRAMACION BASIC forman la subcadena PROGR.


IC25
R44
VID
R56
R39
C51
C4
R1
R9
IC10
IC8
C64
C63
R70
IC9
IC7

3
Programación del ordenador

3.0
Introducción

Para un actor, ¿qué diferencia hay entre hacer una obra de teatro y hacer una película? La respuesta es que en una película la ejecución del actor es almacenada en "película" o en cinta para ser reproducida y vista más tarde, pero el teatro en vivo es "aquí y ahora". Una vez que el actor dice sus líneas, ese momento se hace historia. La comparación teatro-película es interesante para la comprensión de los dos modos en que un ordenador puede operar. En el *modo inmediato* las instrucciones para el ordenador son ejecutadas "en vivo". Para ser reproducidas otra vez deben ser introducidas de nuevo en el ordenador. Este ha sido el modo que se ha venido empleando en los capítulos 1 y 2 (afortunadamente usted ha estado trabajando con un ordenador desde el principio al fin y ha visto los resultados "inmediatos"). Pero en el *modo diferido,* el ordenador guarda las instrucciones que recibe, de tal manera que pueden ser ejecutadas más tarde, cuando se le ordene. Una serie de instrucciones introducidas con el *modo diferido* se llama "programa". Las instrucciones de un programa pue-

den ser utilizadas y reutilizadas igual que una película. E igual que una película puede ser alterada después de que es filmada, un programa puede ser alterado después de haber sido introducido. El poder real de un ordenador es su habilidad para operar en *modo diferido.*

Es una buena idea leer ahora el apéndice A, ya que le instruirá acerca del Sistema Operativo de Discos de tal manera que usted puede conservar los programas que realice.

3.1

Las líneas numeradas y la orden RUN

No hay conmutador o botón alguno para poner al ordenador en el *modo diferido.* Entonces, ¿cómo podemos hacerlo? La respuesta está en las *líneas numeradas.* Si introducimos:

```
10 PRINT 2+3
```

la línea numerada 10 le informa al ordenador que esta instrucción PRINT *no* es para ser ejecutada ahora, sino para ser almacenada para un uso posterior (la sentencia es conservada en la memoria del ordenador, pero no exactamente de la misma manera que los datos son almacenados en las variables).

La orden que manda al ordenador ejecutar las sentencias de la memoria es RUN. Se introduce *sin* número de línea, así que su efecto es inmediato. En la pantalla veremos:

```
10 PRINT 2+3

]RUN
5
```

Cuando son introducidas varias sentencias en el ordenador en *modo diferido,* sus números de línea deben ser diferentes. Cuando sea dada la orden RUN, el ordenador ejecutará las sentencias *en orden numérico.* Por ejemplo:

```
10 PRINT "PREPARADOS"
20 PRINT "LISTOS"
30 PRINT "¡YA!"

]RUN
PREPARADOS
LISTOS
¡YA!
```

Incluso si las sentencias han sido introducidas así:

```
30 PRINT "¡YA!"
10 PRINT "PREPARADOS"
20 PRINT "LISTOS"
```

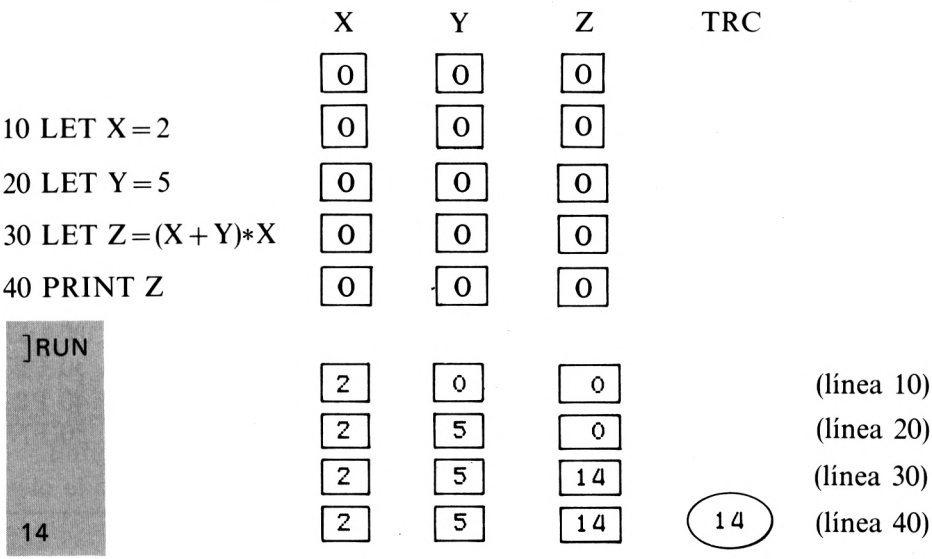
La respuesta sería la misma que antes. El ordenador ejecuta las sentencias en orden numérico. Los números de línea dicen al ordenador en qué orden deben ser ejecutadas las sentencias.

NORMAS PARA LOS NUMEROS DE LINEA

1. En el Applesoft, los números de línea pueden ser cualquier número entero de 0 a 63999 (de 1 a 9999 en la mayoría de los otros BASIC).
2. Los números de línea determinan el orden en que las sentencias son ejecutadas: desde el menor número al mayor.
3. Se recomienda que los números de línea no sean correlativos, por ejemplo, 10, 20, 30, etc., con la finalidad de poder insertar otras líneas entre las existentes cuando posteriormente se quiera alterar el programa.

Echemos una mirada a otro programa. Su diagrama de cajas esta representado a la derecha de él:

```
10 LET X=2
20 LET Y=5
30 LET Z=(X+Y)*X
40 PRINT Z
```



Nótese que las cajas X, Y y Z no son alteradas hasta que la orden RUN es escrita. Después el ordenador procesa las sentencias secuencialmente, como muestra el diagrama de cajas. Desde ahora en adelante, sin embargo, siempre que un diagrama de cajas sea ofrecido junto a un

programa, el efecto de cada una de las líneas será mostrado simultáneamente en las cajas. Continúe teniendo en mente, por supuesto, que el efecto real sobre las cajas no tiene lugar hasta que la orden RUN es ejecutada. El programa anterior podría, entonces, representarse de esta manera:

	X	Y	Z	TRC
	0	0	0	
10 LET X=2	2	0	0	
20 LET Y=5	2	5	0	
30 LET Z=(X+Y)*X	2	5	14	
40 PRINT Z	2	5	14	14

3.2

La instrucción END

La instrucción END es una instrucción BASIC que manda al ordenador que interrumpa la ejecución de las instrucciones en el *modo diferido.* En el Applesoft, el proceso de las sentencias en modo diferido se para automáticamente cuando se alcanza la última sentencia. Por esta razón, una sentencia END, como última línea de un programa, es opcional (algunas versiones de BASIC, sin embargo, requieren que un programa termine con una sentencia END). Nosotros recomendamos que cada programa BASIC termine con una sentencia END aunque sea opcional. La razón se verá claramente cuando lea lo que sigue acerca de la orden LIST.

Ejemplo Los tres programas de la última sección pueden (y deben, es nuestra recomendación) ser escritos de esta manera:

```
10 PRINT 2+3
20 END
```

```
10 PRINT "PREPARADO"
20 PRINT "LISTO"
30 PRINT "¡YA!"
40 END
```

```
10 LET X=2
20 LET Y=5
30 LET Z=(X+Y)*X
40 PRINT Z
50 END
```

3.3

La orden LIST y la edición

La introducción de la orden LIST en *modo inmediato* (esto es, sin un número de línea) hace al ordenador visualizar todas las líneas de sentencia numeradas que han sido almacenadas en la memoria. LIST siempre mues-

tra las sentencias de un programa en orden numérico, aunque las sentencias hayan sido introducidas sin orden.

Suponga que ha introducido este programa:

```
10 PRINT "TRES"
20 PRINT "DOS"
30 PRINT "IGNICION!"
40 END
```

y olvidó incluir PRINT "UNO". Simplemente introduzca

```
25 PRINT "UNO"
```

para insertar la línea, y entonces introduzca LIST para ver qué ha ocurrido:

```
]LIST
10 PRINT "TRES"
20 PRINT "DOS"
25 PRINT "UNO"
30 PRINT "IGNICION!"
40 END
```

Ahora se ve por qué es bueno dejar intervalos entre los números de las líneas. No habría sido posible insertar una línea entre dos sentencias numeradas 2 y 3.

Observación: Para reemplazar una línea, basta reescribir dicha línea. Por ejemplo, si queremos la línea 30 para decir PRINT "¡YA!" sólo necesitamos reintroducirla:

```
]30 PRINT "¡YA!"
]LIST
10 PRINT "TRES"
20 PRINT "DOS"
25 PRINT "UNO"
30 PRINT "¡YA!"
40 END
```

(Las líneas pueden también ser alteradas sin ser reintroducidas completamente. Sin embargo, la técnica es algo complicada. Ver el apéndice D.)

Para eliminar completamente una línea de un programa basta escribir sólo el número de la línea y presionar la tecla RETURN, así:

```
]30
]LIST
10 PRINT "TRES"
20 PRINT "DOS"
25 PRINT "UNO"
40 END
```

Un grupo de líneas puede ser eliminado de una vez usando la orden de borrado, llamada DEL, como se indica:

```
]DEL 10,25
]LIST
40 END
```

La orden DEL puede tener otras formas en otros ordenadores. Anteriormente recomendábamos que la sentencia END fuera usada como última sentencia de un programa, aunque ésta sea opcional. La razón es que como su programa sufrirá frecuentes alteraciones, las líneas pueden ser inadvertidamente borradas. Si un LIST completo de su programa no termina con una sentencia END, entonces sabrá que el programa ha sido alterado. No se preocupe, tales casos son raros. Pero merece la pena el pequeño esfuerzo que supone introducir una sentencia END.

3.4

Las órdenes NEW y CLEAR

Antes de comenzar a introducir un nuevo programa en el ordenador debe introducirse siempre la orden NEW en *modo inmediato*. La orden NEW vacía la memoria del ordenador de todas las instrucciones previamente almacenadas en *modo diferido* y también vacía el contenido de todas las variables.

Las variables numéricas son puestas en cero, mientras que las variables de cadena son convertidas en "cadena nula" (que es una cadena de longitud cero). Es muy importante acordarse de usar la orden NEW, ya que de no hacerlo podemos encontrarnos líneas inesperadas cuando ordenemos un LIST en nuestro programa. *Observación:* No es una buena idea usar NEW como una sentencia en *modo diferido* dentro de un programa, ¡a menos que la intención sea tener un programa que se borre a sí mismo en plena ejecución!

Si sólo se quiere inicializar todas las variables pero conservar intacto su programa, se utilizará CLEAR, que puede usarse tanto con el *modo inmediato* como con el *modo diferido*.

3.4

Ejercicios

1. Escriba programas BASIC para las siguientes secuencias de instrucciones BASIC. Use la sentencia END. Si las va a utilizar en un ordenador asegúrese de introducir NEW antes de introducir las líneas del programa.

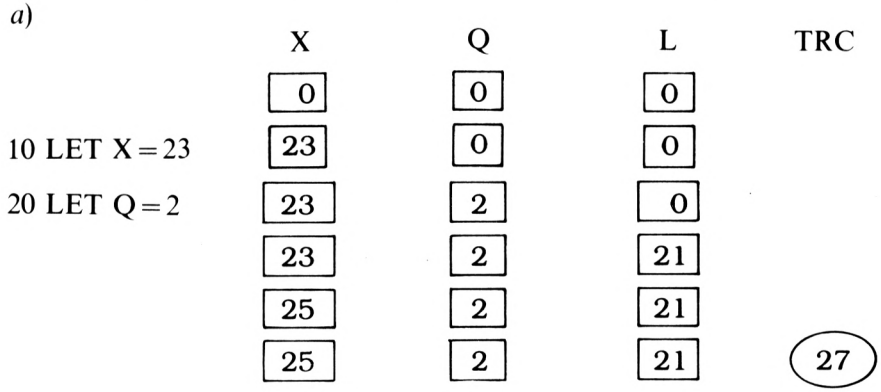
a)
```
LET P=126
PRINT "UNA PERSONA QUE PESE"
PRINT P
PRINT "KILOS EN LA TIERRA"
PRINT "PESARA"
PRINT P/6
PRINT "KILOS EN LA LUNA"
```

b)
```
LET T1=80
LET T2=85
LET T3=93
LET A=(T1+T2+T3)/3
PRINT "MI NOTA MEDIA"
PRINT A
```

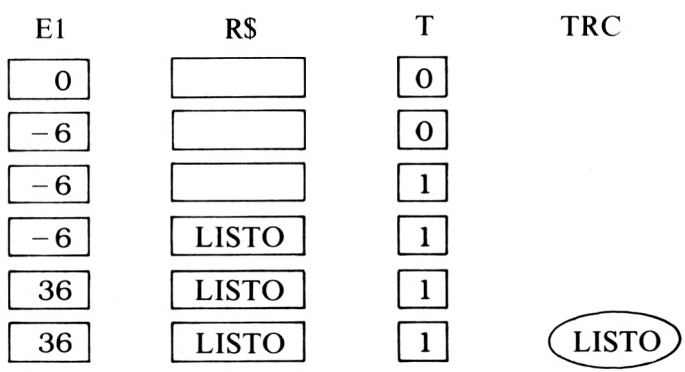
2. Dibuje un diagrama de cajas para el problema 1.

3. Escriba un programa BASIC por cada diagrama de cajas siguiente. A ser posible, utilice sólo variables en las sentencias que escriba.

a)



b)



A continuación describiremos algunas aplicaciones elementales de programación del ordenador.

APLICACION: BALANCE DEL TALONARIO DE CHEQUES

El siguiente programa BASIC hace el balance del talonario de cheques de la señora Jérez, suponiendo que el saldo anterior de su cuenta corriente es de 45885 pesetas, que el total de cheques cobrados es 4576 pesetas y el total de ingreso es de 12550 pesetas.

```
100 LET B=45885
110 LET D=12550
120 LET C=4576
130 LET S=B+D-C
140 PRINT S
150 END
```

```
]RUN
53839
```

3.4

Ejercicios (cont.)

4. Escriba un diagrama de cajas para el programa del talonario de cheques anterior.

5. Suponga que intercambiamos las instrucciones de las líneas 100 y 130 como sigue:

```
100 LET S=B+D-C
130 LET B=45885
```

Escriba un nuevo diagrama de cajas. ¿Funciona el programa correctamente? Explíquelo.

APLICACION: CALCULO DE UN ANGULO DE UN TRIANGULO

En geometría, es bien sabido que la suma de los ángulos de un triángulo es 180 grados. Si A, B y C representan los ángulos de un triángulo, podemos decir que $A+B+C=180$. Si dos de los ángulos son conocidos, el tercer ángulo A puede ser calculado por la fórmula:

$A=180-B-C$

El siguiente programa BASIC calcula A, dados los valores de B y C:

```
100 LET B=34
110 LET C=110
120 LET A=180-B-C
130 PRINT A
140 END
```

```
]RUN
36
```

3.4

Ejercicios (cont.)

6. Construya un diagrama de cajas para el programa anterior.

7. Intercambie las instrucciones de las líneas 100 y 120. Construya un nuevo diagrama de cajas. Explique el efecto que tendrá el intercambio sobre el resultado.

8. Todos los triángulos rectángulos tienen un ángulo de 90°. Suponga que los otros dos ángulos son A y B. Escriba un programa BASIC para calcular el ángulo B, siendo el ángulo A igual a 26.72°. Suponga que el triángulo es rectángulo.

APLICACION: CALCULO DE INTERESES

El señor Montes compró un reloj por 6000 pesetas. Pagó 1000 pesetas al contado y aplazó el resto con un interés de 1.5 por 100 por mes. El siguiente programa BASIC calculará el interés total adeudado al final del primer mes:

```
100 LET P=6000
110 LET D=1000
120 LET R=.015
130 LET I=(P-D)*R
140 PRINT I
150 END
```

```
]RUN
7500
```

3.4

Ejercicios (cont.)

9. Diseñe un diagrama de cajas para el programa anterior.

10. Suponga que el señor Montes pagó 250 pesetas como pago del primer mes. Modifique el programa anterior para calcular también el total que adeuda del reloj después del primer mes.

APLICACION: INTERESES OBTENIDOS

El señor Flores invirtió 5000 pesetas en una cuenta de ahorro al 1.5 por 100 por mes. El siguiente programa BASIC calculará el interés obtenido después del primer mes y el capital total en la cuenta.

```
 5 LET P=5000
10 LET R=.015
15 LET I=P*R
20 LET P=P+I
25 PRINT "INTERES OBTENIDO"
30 PRINT I
35 PRINT "CAPITAL TOTAL DESPUES DE UN MES"
40 PRINT P
45 END
```

```
]RUN
INTERESES OBTENIDOS
7500
CAPITAL TOTAL DESPUES DE UN MES
507500
```

3.4

Ejercicios (cont.)

11. Construya un diagrama de cajas para el problema anterior.

12. Modifique el programa para una inversión inicial de 4800 pesetas a un interés del 2 por 100 al mes.

13. Modifique el programa del problema 12 para calcular e imprimir en pantalla el total de los intereses ganados y el total del capital en dos meses.

APLICACION: ANALISIS DE LAS EXISTENCIAS DE MERCADO

El señor Rubio compró 120 acciones de IBM a 76.75 dólares cada acción, y 76 acciones de ITT por 23.50 dólares cada acción. El siguiente programa BASIC calculará e imprimirá en la pantalla su inversión total

```
100 LET IBM = 120
200 LET P1 = 76.75
300 LET ITT = 76
500 LET P2 = 23.50
550 LET T = IBM*P1 + ITT*P2
575 PRINT "LA INVERSION TOTAL DEL SR. RUBIO ES"
600 PRINT T
800 END
```

```
]RUN
INVERSION TOTAL DEL SR. RUBIO
1099600
```

3.4 Ejercicios (cont.)

14. Construya un diagrama de cajas para el programa anterior.

15. Suponga que el señor Rubio también compró 250 acciones de Ford Motors a 65.25 dólares cada acción. Modifique el programa anterior para calcular e imprimir en pantalla su inversión total.

3.5 La instrucción REMark

La instrucción REM es una de las varias sentencias no ejecutables en BASIC. Cuando el ordenador encuentra la instrucción REM en un programa, la ignorará y continuará con la siguiente instrucción de la secuencia. REM quiere decir REMark (comentario) y se utiliza para documentar un programa. Es útil para colocar comentarios en el programa, tales como: quién escribió el programa y cuándo, lo que el programa se supone que hace, lo que significa cierta fórmula, y qué variables están siendo usadas. Las sentencias REM pueden verse sólo cuando se hace un listado del programa.

Ejemplo

APLICACION: CONVERTIR AÑOS EN MINUTOS

```
10 REM PROGRAMA PARA CONVERTIR CINCO AÑOS EN
   MINUTOS
20 REM ESCRITO POR JUAN Q., ESTUDIANTE
30 REM 31, DIC. 1983
40 REM
```

```
 50 REM LISTA DE VARIABLES
 60 REM A ES EL NUMERO DE AÑOS
 63 REM D ES EL NUMERO DE DIAS
 65 REM H ES EL NUMERO DE HORAS
 67 REM M ES EL NUMERO DE MINUTOS
 69 REM
 70 LET A=5
 80 LET D=365*A
 90 LET H=24*D
100 REM LAS HORAS YA HAN SIDO CALCULADAS, AHORA
    OBTENDREMOS LOS MINUTOS
120 LET M=60*H
130 PRINT "EL NUMERO DE MINUTOS ES"
140 PRINT M
150 END
```

3.5

Ejercicios

1. Realice un diagrama de cajas para el programa anterior.

2. Escriba un programa que convierta los minutos en años.

3.6

Instrucciones múltiples en una sola línea

Poner varias instrucciones en una línea del programa es posible en el Applesoft, y en muchas otras versiones de BASIC. En el Applesoft, se usan dos puntos (:) para separar las instrucciones.

Ejemplo Las líneas 70, 80 y 90 del programa anterior pueden ser escritas como una línea, así:

```
70 LET A=5 : LET D=365*A : LET H=24*D
```

Las desventajas de las líneas de instrucciones múltiples generalmente superan a las ventajas. Por un lado, insertar una nueva instrucción es difícil si debe ir entre dos instrucciones de una misma línea. También, una instrucción REM no puede ser colocada en una línea de instrucciones múltiples a menos que lo haga al final de la línea.

Esto es porque los dos puntos (:) que siguen a REM son considerados parte del comentario.

Ejemplo La instrucción

```
100 REM SUME DOS NUMEROS : LET X=Y+Z
```

es estrictamente una REM. No hay instrucción LET, ya que cualquier cosa que vaya después de REM es ignorada.

La sentencia

```
100 LET X=Y+Z : REM SUME DOS NUMEROS
```

es una instrucción LET seguida de una REM.

Las ventajas de usar instrucciones múltiples se verán en los capítulos 6 y 8.

Excepto para situaciones especiales en que las líneas de sentencias múltiples son ventajosas, para mayor claridad, nosotros empleamos líneas de sentencias simples.

3.7

Programación estructurada: método de módulos ACS

Para ser un buen programador BASIC usted debe aprender un método para organizar las instrucciones.

El método ACS divide el programa BASIC en tres módulos indicados por las letras, A, C y S:

I. *Asignación:* El primer módulo se llama *de asignación.* En este módulo los datos son asignados a las variables.

II. *Cálculo:* El segundo módulo se llama *de cálculo.* En este módulo son calculados los valores.

III. *Salida:* El tercer módulo se llama *de salida.* En este módulo los datos son visualizados en la pantalla.

Para programas simples, una buena norma a seguir es: *escribir primero el módulo de asignación, segundo el módulo de cálculo y tercero el módulo de salida.*

En los últimos capítulos ampliaremos el concepto de módulo para construir programas más complejos. Estudie los ejemplos que siguen.

Ejemplo

```
10 REM DEMOSTRACION DE METODO DE MODULOS ACS
20 REM
30 REM ESTE PROGRAMA HALLA LA HIPOTENUSA DE
40 REM UN TRIANGULO RECTANGULO CUYOS LADOS SON
   CONOCIDOS
```

```
 50 REM
 60 REM MODULO DE ASIGNACION
 65 REM
 70 LET A=6
 80 LET B=8
 90 REM
100 REM MODULO DE CALCULO
105 REM
110 REM ESTE ES EL TEOREMA DE PITAGORAS
120 LET C=SQR(A^2+B^2)
130 REM
140 REM MODULO DE SALIDA
150 REM
160 PRINT "LA HIPOTENUSA ES IGUAL A"
170 PRINT C
180 END
```

Ejemplo El siguiente programa debe serle familiar. Es el programa de los *intereses obtenidos* organizado usando el método ACS.

```
100 REM INTERESES OBTENIDOS POR EL SR. FLORES
110 REM
120 REM MODULO DE ASIGNACION
130 LET P=500,000
140 LET R=.015
150 REM
160 REM MODULO DE CALCULO
170 LET I=P*R
180 LET P=P+I
190 REM
200 REM MODULO DE SALIDA
203 REM
205 PRINT "INTERESES OBTENIDOS"
210 PRINT I
215 PRINT "CAPITAL DESPUES DE UN MES"
220 PRINT P
240 END
```

Debe decirse que el método ACS es un conjunto de orientaciones, no una norma fija. *Utilice el método de módulos ACS tan amplia y extensamente como sea posible.* Le ayudará a organizar sus pensamientos y a escribir lógicamente programas complejos.

3.7

Ejercicios 1. Reescriba los siguientes programas empleando el método ACS:

a)
```
20 LET X=24
30 LET Y=-4
40 PRINT X
50 LET Z=X+Y
60 PRINT Z
70 LET T=2*X-Y^3
80 PRINT T
90 END
```

b)
```
10 LET Q$="MI"
15 PRINT Q$
20 LET R$="NOMBRE"
30 PRINT R$
40 LET E1$="ES"
50 PRINT E1$
60 LET W$="MOZART"
80 PRINT W$
90 PRINT "!"
100 END
```

3.8

Proyectos

Escriba como programas BASIC los proyectos dados en el capítulo 1.

PRACTICA 3.1

Los modos inmediato y diferido

Introduzca la orden: NEW

El ordenador ejecuta las instrucciones en dos diferentes modos, el modo inmediato y el modo diferido.

Las instrucciones que son ejecutadas con el modo inmediato se llaman *órdenes,* y las instrucciones que son ejecutadas con el modo diferido se llaman *sentencias.* Las sentencias tienen números de línea, las órdenes no. Siempre que una sentencia es introducida en el ordenador, éste la almacenará en la memoria y retrasará su ejecución hasta que la orden RUN le sea introducida. Siempre que se introduzca una orden en el ordenador, éste la ejecutará inmediatamente.

Ciertas instrucciones sólo se pueden usar como órdenes, mientras que otras pueden usarse como sentencias. La mayor parte de las instrucciones, sin embargo, ¡pueden usarse de las dos maneras! Por ejemplo, LET y PRINT pueden ser usadas como órdenes o como sentencias.

Introduzca la sentencia: 10 LET X=5

El número 10 se llama número de línea de la sentencia.

Introduzca la orden: PRINT X

¿Cuál es la respuesta del ordenador? _____

Explique por qué el ordenador imprimió 0 en vez de 5 en la pantalla. _____

_____

_____

La razón es que el ordenador no ejecutó la sentencia 10 LET X=5. No ejecutará esta orden hasta que sea introducida la orden RUN.

Introduzca la orden: RUN

¿Qué valor espera usted que contenga ahora la variable X? _____

Introduzca la orden: PRINT X

¿Cuál es la respuesta del ordenador? _____

Explique por qué la respuesta es 5. _____

_____

_____

Después de que es introducida la orden RUN, se ejecuta la sentencia 10 LET X=5, asignando el número 5 a la variable X. Así, 5 pudo imprimirse en la pantalla cuando fue introducida la orden PRINT X.

Siempre que es ejecutada la orden RUN, el ordenador inicializa todas las variables que tienen almacenadas en memoria. Es decir, todas las variables numéricas son puestas a cero y todas las variables de cadena son convertidas en "variables nulas" (cadenas de longitud cero).

Introduzca la orden: LET Y=7

¿Qué valor es almacenado en la variable Y? _____

Ya que esto es una orden, el ordenador asigna el número 7 a Y.

Escriba la orden: RUN

¿Qué valor es ahora almacenado en la variable Y? _____

Explique por qué el ordenador mostró 0 en vez de 7. _____

_____

_____

_____

Al ejecutarse la orden RUN, el ordenador inicializó las variables. Así, a Y le fue asignado el valor 0.

PRACTICA 3.2

LIST y DEL

Introduzca la orden: NEW

Introduzca las sentencias: 10 LET X=5
20 PRINT X

Introduzca la orden: HOME

Como puede observarse, la orden HOME limpió la pantalla, ¡nada de pánico! Su programa está aún en la memoria.

Para hacer que el ordenador muestre las sentencias que están almacenadas, se usa la orden LIST.

Introduzca la orden: LIST

¿Cuál es la respuesta del ordenador? _____

_____

_____

La orden LIST visualizará las instrucciones almacenadas en la memoria, en orden numérico.

Introduzca las sentencias: 30 PRINT Y
15 LET Y=8

Introduzca la orden: LIST

¿Cuál es la respuesta del ordenador? _____

_____

_____

No importa en qué orden se introduzcan las sentencias de un programa. El listado (LIST) y ejecución del programa siempre se harán según el orden de los números de línea.

La orden LIST se puede usar para mostrar una o más líneas numeradas.

Introduzca la orden: LIST 15

¿Cuál es la respuesta del ordenador? _____

La respuesta es 15 LET Y=8. Al presentar todo el rango de números de línea, los argumentos de la orden LIST son el primer y último número de línea separados por una coma.

Introduzca la orden: LIST 10,20

¿Qué líneas numeradas son mostradas por el ordenador? _____

Para cambiar una sentencia de un programa, introduzca la nueva sentencia con el mismo número de línea. Por ejemplo, cambiemos la instrucción 15 LET Y=8*X.

Introduzca la sentencia: 15 LET Y=8*X

Introduzca la orden: LIST

¿Qué cambio observa usted en el programa? _____

_____

Alterar puede requerir también la eliminación de una sentencia del programa. Para eliminar una sentencia, introduzca el número de línea de la sentencia.

Escriba: 20 (y pulse la tecla RETURN)

Introduzca la orden: LIST

¿Cuál es la respuesta del ordenador? _____

_____

_____

¿Qué sentencia se borró del programa? _____

La sentencia 20 PRINT X fue borrada del programa.

Introduzca las sentencias: 40 PRINT X
50 LET A=2
60 PRINT X+A

Introduzca la orden: LIST

¿Cuál es la respuesta del ordenador? 10 _____

15 _____

30 _____

40 _____

50 _____

60 _____

Para borrar un intervalo de líneas se usa la orden DEL.

Introduzca la orden: DEL 15,40

Introduzca la orden: LIST

La orden DEL requiere dos números separados por una coma. Los números son el primero y el último de los números de línea del intervalo que quiere ser borrado.

¿Qué números de línea fueron borrados del programa? _____

Usted debe haber escrito los números de línea 15, 30 y 40. Observe que estos números de línea están entre el 15 y el 40, inclusive.

PRACTICA 3.3

Estructuración de programas utilizando el método ACS

Introduzca la orden: NEW

A un conjunto de sentencias introducidas en "modo diferido" se les llama *programa.* Las sentencias de un programa son ejecutadas por el ordenador de una en una. El ordenador ejecuta estas sentencias en el orden secuencial de los números de línea.

El orden lógico de las sentencias de un programa es de la mayor importancia.

Introduzca las sentencias:
```
10 LET F=9/5*C+32
20 LET C=20
30 PRINT F
```

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

La intención de este programa es convertir 20 grados Celsius en grados Fahrenheit (20 grados Celsius es equivalente a 68 grados Fahrenheit).

Explique por qué el ordenador visualizó 32 en vez de 68. _____

_____

_____

El ordenador ejecutará el programa en el orden numérico de los números de línea. Primero se ejecuta la línea 10, asignando a F el valor calculado de 9/5*C+32. Así, el valor calculado de 9/5*C+32 es 32. A C le es asignado el valor 20 *después* de la asignación de F.

Introduzca las sentencias:
```
10 LET C=20
20 LET F=9/5*C+32
```

Introduzca la orden: LIST

¿Qué diferencia hay con el programa anterior? _____

_____

Los números de línea 10 y 20 están intercambiados.

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

La primera sentencia ejecutada en este programa es 10 LET C=20. Después de que 20 ha sido asignado a C, se realiza la ejecución de la sentencia 20 LET F=9/5*C+32.

La mayoría de los programas tienen tres partes: 1. Asignación.
2. Cálculo.
3. Salida.

Siempre que se escribe un programa se hace primero un esquema. El esquema ayuda al programador a escribir el programa en un orden lógico. Por ejemplo, el esquema de un programa que evalúe y escriba todas las sumas posibles de tres números dados puede ser éste:

1. Asignar el primer número a X.
2. Asignar el segundo número a Y.
3. Asignar al tercer número a Z.
4. Evaluar la suma de X más Y y almacenarla en A.
5. Evaluar la suma de X más Z y almacenarla en B.
6. Evaluar la suma de Y más Z y almacenarla en C.
7. Evaluar la suma de X, más Y más Z y almacenarla en D.
8. Imprimir A en la pantalla.
9. Imprimir B.
10. Imprimir C.
11. Imprimir D.

Usando el esquema anterior, escriba un programa que imprima todas las sumas posibles de los números 2, 5 y 8.

10 _____

20 _____

30 _____

40 _____

50 _____

60

70

80

90

100

110

120

Su programa podría ser el siguiente:

Introduzca el programa:

```
 5 REM MODULO DE ASIGNACION
 6 REM
10 LET X=2
20 LET Y=5
30 LET Z=8
34 REM
35 REM MODULO DE CALCULO
36 REM
40 LET A=X+Y
50 LET B=X+Z
60 LET C=Y+Z
70 LET D=X+Y+Z
74 REM
75 REM MODULO DE SALIDA
76 REM
80 PRINT A
90 PRINT B
100 PRINT C
110 PRINT D
120 END
```

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador?

Introduzca la orden: NEW

Se pueden combinar los diferentes módulos. Por ejemplo, suponga que hemos escrito el esquema del programa de esta manera:

1. Asignar el primer número a X.
2. Asignar el segundo número a Y.
3. Asignar el tercer número a Z.
4. Evaluar la suma de X más Y e imprimirla.
5. Evaluar la suma de X más Z e imprimirla.
6. Evaluar la suma de Y más Z e imprimirla.
7. Evaluar la suma de X más Y más Z e imprimirla.

Utilizando el esquema, escriba el programa que muestre en la pantalla todas las posibles sumas de los números 2, 5 y 8.

10
_____
20
_____
30
_____
40
_____
50
_____
60
_____
70
_____
80
_____

Su programa podría ser como el que sigue:

Introduzca las sentencias:

```
 5 REM MODULO DE ASIGNACION
10 LET X=2
20 LET Y=5
30 LET Z=8
35 REM MODULO DE CALCULO Y SALIDA
40 PRINT X+Y
50 PRINT X+Z
60 PRINT Y+Z
70 PRINT X+Y+Z
80 END
```

Introduzca la orden: LIST

Fíjese que los módulos de cálculo y salida forman un solo módulo.

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

_____

_____

Aunque el programa se escribió de forma diferente, la respuesta es la misma. El método de módulos ACS debe ser utilizado tanto como sea posible. Sin embargo, no es un método obligatorio para escribir los programas. Nosotros resaltamos aquí que el orden lógico de las sentencias en un programa es crucial.

PRACTICA 3.4

Ordenes de disco

Antes de hacer esta práctica, asegúrese de haber inicializado el disco. Las instrucciones sobre cómo hacer un INIT (inicializar) a los discos se dan en el apéndice A.

Introduzca la orden: NEW

Introduzca el programa:

```
10 REM CONVERTIR CELSIUS A FAHRENHEIT
20 LET C=20
30 LET F=9/5*C+32
40 PRINT F
50 END
```

Para almacenar un programa en disco para usarlo posteriormente se usa la orden SAVE (guardar). El argumento de la orden SAVE es el nombre del programa.

Introduzca la orden: SAVE FAHRENHEIT

Cuando el ordenador ejecuta la orden, una copia del programa que está en la memoria se almacena en el disco. El programa será almacenado en el disco bajo el nombre dado con la orden SAVE.

Introduzca la orden: NEW

Introduzca el programa:

```
10 REM CONVERTIR FAHRENHEIT A CELSIUS
20 LET F=50
30 LET C=5/9*(F-32)
40 PRINT C
50 END
```

Introduzca las órdenes: SAVE CELSIUS
LIST

Fíjese que el programa no se borró de la memoria después de que el programa fue guardado con SAVE. Recuerde, el ordenador hace una copia del programa que está en la memoria y lo almacena en el disco.

La orden CATALOG hará un listado de todos los nombres de los programas que usted haya almacenado en el disco.

Introduzca la orden: CATALOG

Usted debe ver los nombres de los tres programas que se guardaron: HELLO, FAHRENHEIT y CELSIUS. El programa HELLO fue creado y guardado cuando inicializó el disco.

Para recuperar un programa almacenado en su disco se usa la orden LOAD. El argumento de la orden LOAD es el nombre del programa.

Introduzca las órdenes: LOAD FAHRENHEIT
LIST

¿Qué programa está ahora almacenado en la memoria del ordenador? _____

_____

Fíjese que CELSIUS fue borrado de la memoria, y FAHRENHEIT está ahora almacenado en ella. La orden LOAD limpiará la memoria, recuperará el programa que usted quería del disco, y lo almacenará en ella.

Introduzca la orden: SAVE CELSIUS

Guardar un programa usando el mismo nombre de un programa ya almacenado en el disco obligará al ordenador a reemplazar el programa almacenado en el disco por el programa de la memoria.

Introduzca la orden: LOAD CELSIUS

Introduzca la orden: LIST

¿Son Fahrenheit y Celsius el mismo programa? _____

_____

Su respuesta debe ser SI.

Explique por qué ocurre esto. _____

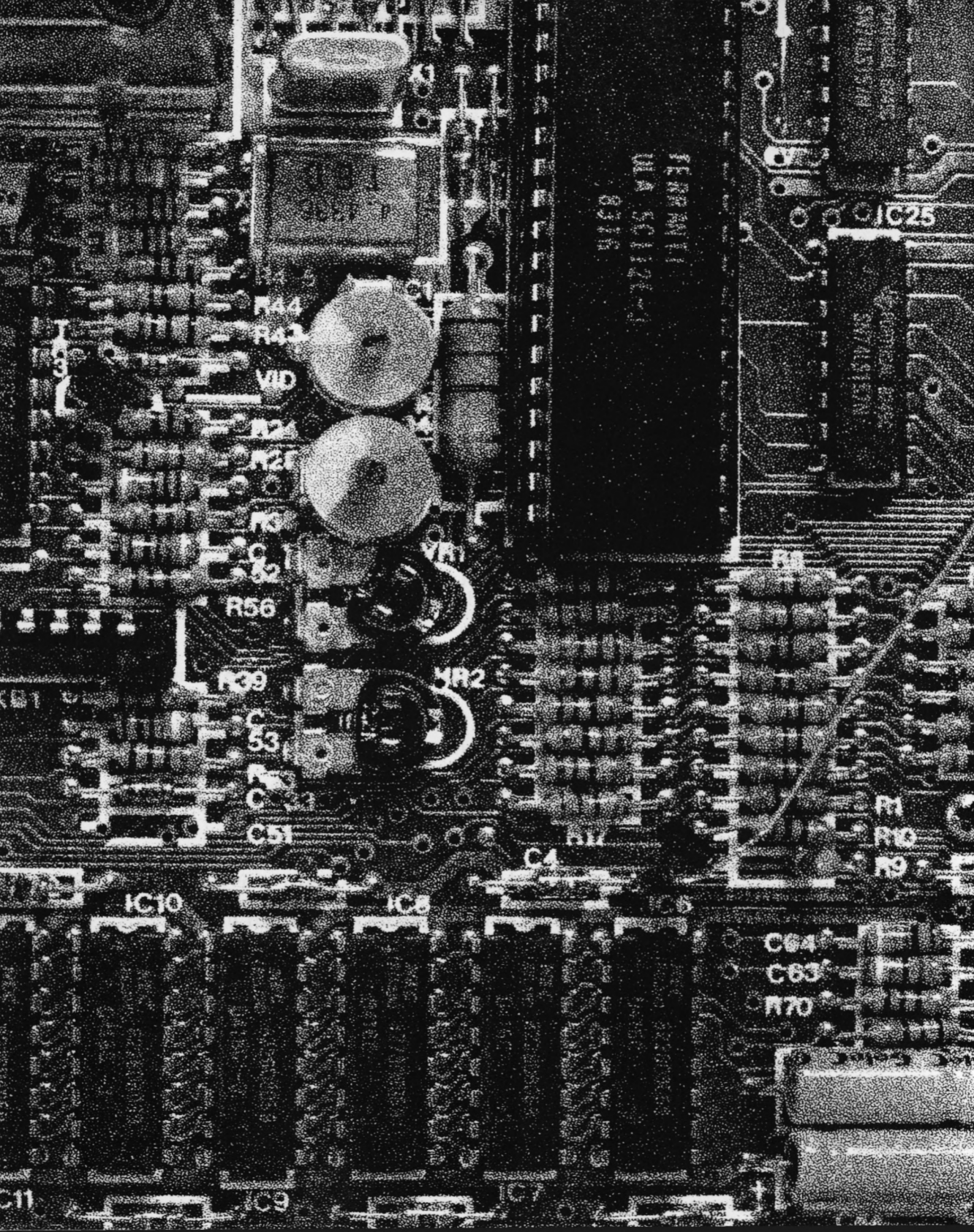
_____

Para borrar un programa del disco, se emplea la orden DELETE (borrar). El argumento de la orden DELETE es el nombre del programa.

Introduzca las órdenes: DELETE CELSIUS
CATALOG

¿Qué programa ya no está en el disco? _____

_____


IC25
R44
VID
R56
R39
C51
C4
R1
R9
IC10
IC8
C64
C63
R70
IC9
IC7

4
Más sobre la impresión

4.0
Introducción

En el capítulo 1 se trató la orden PRINT en su forma más simple. Aquí discutiremos tres características adicionales de la instrucción PRINT que pueden usarse para hacer más vistoso lo mostrado en la pantalla. Son la coma, el punto y coma y TAB(X). Para las instrucciones adicionales Applesoft que afectan a la aparición sobre la pantalla, ver el apartado del capítulo 12 llamada "presentación en la escritura".

4.1
La coma

En el Applesoft II BASIC, cada línea de la pantalla está dividida en tres invisibles zonas de impresión; algo así como los espacios fijos tabulados en una máquina de escribir. Si se usa una coma para separar los

ítems (elementos) de una sentencia PRINT, la impresión comenzará en la primera zona libre de la línea. Si no hay más zonas para imprimir en la línea, la impresión continuará en la siguiente línea de la pantalla.

Las zonas de impresión comienzan en la posición del carácter 1, 17 y 33 de la línea. Ya que el Apple muestra texto sobre una pantalla de cuarenta caracteres, las zonas de impresión no son de igual longitud. Las dos primeras son de dieciséis caracteres de largo, y la tercera es de ocho. Hay que tener en cuenta que otras versiones del BASIC tendrán formatos de zonas de impresión diferentes, aunque la función de la coma continúe siendo la misma.

Ejemplo

```
10 PRINT "1234567890123456789012345678901234567890"
20 PRINT "UNO","DOS","TRES"
30 END
```

```
]RUN
1234567890123456789012345678901234567890
UNO              DOS              TRES
```

Los números están para ayudarle a contar las posiciones de los caracteres. *Importante:* La línea en blanco que hay debajo de la línea numerada está ahí porque la línea anterior se llenó. Se origina una línea nueva automáticamente cuando el Applesoft imprime el carácter de la posición número cuarenta.

Ejemplo

```
10 PRINT "1234567890123456789012345678901234567890"
20 PRINT "HOLA, ¿COMO ESTA USTED?","¡BIEN!"
30 END
```

```
]RUN
1234567890123456789012345678901234567890
HOLA, ¿COMO ESTA USTED?          ¡BIEN!
```

Fíjese que el primer ítem (elemento) se introdujo en la segunda zona de impresión, así que el segundo ítem impreso comenzó en la tercera zona.

Ejemplo

```
10 PRINT "1234567890123456789012345678901234567890"
20 PRINT "PROBANDO","UNO","DOS","TRES","CUATRO"
30 END
```

```
]RUN
1234567890123456789012345678901234567890
PROBANDO         UNO              DOS
TRES             CUATRO
```

Hay más ítems que zonas de impresión, así que la impresión va automáticamente a la línea siguiente.

Ejemplo

```
10 PRINT "1234567890123456789012345678901234567890"
20 PRINT ,,"TINTINABULACION"
30 END
```

```
]RUN

1234567890123456789012345678901234567890

                                    TINTINAB
ULACION
```

Dos cosas deben observarse en este ejemplo. Primera, las dos comas antes de la cadena obligan a que la impresión se haga en la tercera zona. Segunda, ya que la cadena a imprimir es demasiado larga para ocupar los ocho espacios de la última zona de impresión, ésta continúa al otro lado de la pantalla. Esto es cierto en el Applesoft, pero puede no ocurrir en otras versiones del BASIC.

Ejemplo

```
10 PRINT "1234567890123456789012345678901234567890"
20 PRINT "PROBANDO",
30 PRINT "UNO","DOS"
40 END
```

```
]RUN
1234567890123456789012345678901234567890

PROBANDO        UNO                 DOS
```

Fíjese en la coma que hay después de la cadena PROBANDO. Cuando hay una coma al final de una sentencia PRINT, los ítems impresos por la instrucción PRINT subsiguiente comenzarán en la siguiente zona de impresión disponible, justo como si las dos instrucciones PRINT hubieran sido combinadas en una.

4.2 El punto y coma

El efecto del punto y coma es similar al de la coma, excepto que en lugar de decir "zona de impresión contigua" debemos decir "posición de carácter contigua". Otra manera de decir esto es que cuando un ítem

sigue a un punto y coma en una sentencia PRINT, se imprime inmediatamente después del último ítem impreso. Los ejemplos siguientes aclararán esto.

Ejemplo

```
10 PRINT "PASANDO";"DE";"BOCA";"EN";"BOCA"
20 PRINT 1;2;2;3;4
30 END
```

```
]RUN
PASANDODEBOCAENBOCA
12234
```

Ejemplo

```
10 PRINT "PASANDO";"DE";"BOCA";"EN";"BOCA";
20 PRINT 1;2;2;3;4
30 END
```

```
]RUN
PASANDODEBOCAENBOCA12234
```

Esto muestra que si un punto y coma es el último carácter de la línea PRINT, el siguiente dato se imprimirá en la misma línea.

Ejemplo

```
10 PRINT "PASANDO ";"D"DE ";"BOCA ";"EN ";"BOCA "
20 END
```

```
]RUN
PASANDO DE BOCA EN BOCA
```

El efecto de la separación de las palabras se previno al incluir el carácter espacio como parte de las cadenas imprimir.

Recuerde, los contenidos de las variables pueden ser visualizados en pantalla usando la instrucción PRINT. Además las cadenas pueden ser mezcladas dentro de la misma instrucción PRINT.

Ejemplo

```
10 LET X=5
20 LET Y=3
30 LET Z=X-Y
40 PRINT "LA RESPUESTA ES ";Z
50 END
```

```
]RUN
LA RESPUESTA ES 2
```

Ya que se pueden hacer cálculos en una instrucción PRINT, la siguiente secuencia tiene el mismo resultado:

```
10 LET X=5
20 LET Y=3
30 PRINT "LA RESPUESTA ES ";X-Y
40 END
```

```
]RUN
LA RESPUESTA ES 2
```

Ejemplo

```
10 LET N$="SR. LUCAS"
20 PRINT "SI, USTED, SR. LUCAS, YA PUEDE ESTAR PREPARADO"
30 END
```

```
]RUN
SI, USTED, SR. LUCAS, YA PUEDE ESTAR PREPARADO
```

Este último resultado excede los 40 caracteres. En el Apple podríamos ver realmente:

```
]RUN
SI, USTED, SR. LUCAS, YA PUEDE ESTAR PRE-
PARADO
```

4.2 Ejercicios

1. ¿Qué respuesta se verá en la pantalla según lo siguiente?:

a)

```
10 LET X=25
20 PRINT X,SQR(X),X^2
30 PRINT X;SQR(X);X^2
40 PRINT X;"  ";SQR(X);"  ";X^2
50 END
```

b)

```
10 LET N$="SR. TALKER"
20 LET T$="555-8324"
30 PRINT N$;"ES EL TELEFONO DEL";N$
40 END
```

c)

```
10 LET X$="ANTISISTEMARIANISMO"
20 PRINT X$;" TIENE ";LEN(W$) ;" LETRAS."
```

4.3

La función tabulador, TAB(X)

La función TAB(X) es diferente de cualquiera de las funciones que hemos discutido en el capítulo 2. En lugar de dar valores numéricos o de cadena, tiene un efecto físico sobre la posición en que se imprimen ítems en la línea. La función TAB(X) mueve el cursor* N posiciones *desde el margen izquierdo de la pantalla.* TAB(X) sólo puede usarse con una instrucción PRINT.

Ejemplo

```
10 PRINT "12345678901234567890123456789012345678901234567890"
20 PRINT TAB(20);"15, MARZO, 1982"
30 PRINT
40 PRINT "ESTIMADO SR. LUCAS,"
50 PRINT TAB(5);"ESTOS SON SUS PREMIOS: ";TAB(30);"SACO"
60 PRINT TAB(30);"PODADORA"
70 PRINT TAB(30);"EQUIPO"
80 PRINT
90 PRINT TAB(5);"Y UN GRAN PREMIO DE";TAB(30);"25000 PTAS."
95 END
```

```
]RUN
12345678901234567890123456789012345678901234567890

                    15, MARZO, 1982
ESTIMADO SR. LUCAS,
    ESTOS SON SUS PREMIOS:   SACO
                             PODADORA
                             EQUIPO
    Y UN GRAN PREMIO DE       25000
```

Este ejemplo muestra un punto no mencionado anteriormente. Puede crearse una línea en blanco con una sentencia PRINT vacía. Debe saber, sin embargo, que la línea blanca que está debajo de la guía de números de arriba no es debida a una instrucción PRINT. Como se mencionó antes, se genera una línea nueva cuando se ocupa el carácter número 40 de una línea.

Evite las comas en las sentencias PRINT que usen la función TAB(X). Una coma forzaría a los ítems a colocarse en la siguiente zona de impresión, independientemente de donde se haya tabulado. La sentencia

```
PRINT TAB(5),"HOLA"
```

* El cursor es el rectángulo destellante que indica dónde será impreso el próximo carácter.

imprimirá en la posición decimoséptima, no en la quinta. Use siempre el punto y coma, como en

```
PRINT TAB(5);"HOLA"
```

Otra precaución: no use TAB(0), a menos que usted quiera imprimir en la posición 256 del borde izquierdo de la pantalla. ¡Es decir, seis líneas más abajo! La primera posición de una línea es TAB(1).

4.4

Proyectos

1. Escriba un programa para imprimir la siguiente tabla del zodíaco:

SIGNO	PERIODO
CAPRICORNIO	22 DIC A 19 ENE
ACUARIO	20 ENE A 18 FEB
PISCIS	19 FEB A 20 MAR
ARIES	21 MAR A 19 ABR
TAURO	20 ABR A 20 MAY
GEMINIS	21 MAY A 20 JUN
CANCER	21 JUN A 22 JUL
LEO	23 JUL A 22 AGO
VIRGO	23 AGO A 22 SEP
LIBRA	23 SEP A 22 OCT
ESCORPIO	23 OCT A 21 NOV
SAGITARIO	22 NOV A 21 DEC

2. La siguiente tabla muestra las distancias entre las ciudades en millas:

CIUDADES	GOOSE BAY	RAIN HILL	BUTTON RIVER
GOOSE BAY	***	235	400
RAIN HILL	235	***	600
BUTTON RIVER	400	600	***

   Escriba un programa BASIC que imprima esta tabla.

3. Las tablas de funciones matemáticas son muy importantes. Considerar la siguiente tabla:

X	X*X	SQR(X)
1	1	1
2	4	1.414
3	9	1.732
4	16	2
5	25	2.236
6		
7		
8		
9		
10		

Escriba un programa BASIC que complete esta tabla. Redondee la raíz cuadrada a tres lugares de decimales.

4. La señora Romero desea alquilar un coche para conducir desde Los Angeles a San Francisco (400 millas). El Club de Coches Local le provee de la siguiente información de alquiler de coches:

Marca	*Modelo*	*Coste/día (pesetas)*	*Tasa/milla (pesetas)*	*Coste total*
Ford	Escort	2000	23	
Audi	5000	3200	32	
Fiat	Strada	2100	20	
Chevy	Impala	4500	55	
Datsun	210	3100	22	

Escriba un programa que complete la última columna e imprima la tabla entera.

5. El triángulo de Pascal tiene muchas aplicaciones útiles. Las primeras nueve líneas del triángulo son éstas:

```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5   10  10  5   1
    1   6   15  20  15  6   1
  1   7   21  35  35  21  7   1
1   8   28  56  70  56  28  8   1
```

Escriba un programa BASIC que genere e imprima el triángulo de Pascal en la forma mostrada anteriormente. Utilice las sentencias LET para generar cada número a partir de las líneas precedentes. Fíjese que cada número del triángulo es la suma de los dos números que están justo encima de él.

PRACTICA 4.1

El uso de comas en la instrucción PRINT

Introduzca la orden: NEW

La pantalla del APPLE tiene 40 caracteres de ancho que contienen tres zonas de impresión. El uso de comas en una instrucción PRINT colocará los datos dentro de estas zonas. Las primeras dos zonas tienen una longitud de dieciséis (16) caracteres y la tercera zona de ocho caracteres (8). La primera zona va desde la posición del primer carácter hasta 16 inclusive. La segunda zona va desde la posición 17 a la 32 inclusive, y la tercera zona va desde el carácter 33 al 40 inclusive.

Introduzca el programa:
```
10 HOME
20 PRINT "12345678901234567890123456789012345678901234567890"
30 PRINT "A","B","C"
40 END
```

Introduzca la orden: RUN

¿En qué posiciones se imprimen las cadenas A, B, C en el TRC? _____

_____

Las cadenas A, B y C se imprimen en la primera posición de cada zona de impresión (posiciones 1, 17 y 33).

Introduzca la sentencia: 30 PRINT "PROGRAMACION BASIC", 123

Introduzca la orden: LIST

¿Cuántos caracteres hay en la cadena PROGRAMACION BASIC? _____

Introduzca la orden: RUN

¿En qué posiciones aparece 123? _____

Puesto que la cadena PROGRAMACION BASIC tiene 18 caracteres, han sido necesarias dos zonas. Así, 123 se imprimió en la siguiente zona de impresión, es decir, la tercera.

Introduzca la sentencia: 30 PRINT "PROGRAMARA BASIC", 123

Introduzca la orden: LIST

¿Cuántos caracteres hay en la cadena PROGRAMARA BASIC? _____

Introduzca la orden: RUN

¿En qué posición comenzará el número 123? _____

La cadena PROGRAMARA BASIC tiene dieciséis caracteres y se imprime en la primera zona ¡Aún el número 123 está impreso en la tercera zona! El Apple se saltará la segunda zona si se usa la posición 16 de la primera zona. Cuando se usa la posición 24, será saltada la tercera zona.

Introduzca la sentencia: 30 PRINT 123,"PROGRAMA",456

Introduzca la orden: LIST

¿En qué zona aparecerá el número 456 cuando se ejecute el programa? _____

_____

Introduzca la orden: RUN

Puesto que la letra A final de la cadena PROGRAMA ocupa el carácter 24, la tercera zona es saltada. El número 456 se imprime en la siguiente zona de impresión disponible, la cual es la primera de la línea siguiente.

Introduzca las sentencias: 20 PRINT "UNO",
30 PRINT "DOS",
40 PRINT "TRES"

Introduzca la orden: LIST

¿Cuál es el carácter de las líneas numeradas 20 y 30? _____

Introduzca la orden: RUN

La respuesta de este programa es una línea única, aunque haya tres diferentes instrucciones PRINT en el programa.

Siempre que una instrucción PRINT termine con una coma, el dato impreso a continuación lo será en la siguiente zona de impresión.

El uso del punto y coma en la instrucción PRINT

Introduzca la orden: NEW

Siempre que sean usados los punto y comas para separar los datos en una instrucción PRINT, los datos se imprimirán juntos, uno detrás de otro, sin espacios.

Introduzca la orden: PRINT "A","B";"C"

¿Cuál es la respuesta del ordenador? _____

Introduzca el programa:
```
10 LET X$ = "UNO"
20 LET Y = 5
30 PRINT X$;23;"CUATRO";Y
40 END
```

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

La respuesta UNO23CUATRO5 consta de cuatro datos diferentes impresos juntos y sin espacios.

Introduzca las órdenes:
```
30 PRINT X$;
40 PRINT 23;
50 PRINT "CUATRO";
60 PRINT Y
70 END
```

Introduzca la orden: LIST

¿Cuál es el último carácter de las líneas 30, 40 y 50? _____

_____

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

Fíjese que la respuesta es la misma que la del programa anterior. Aunque se ejecutaron diferentes instrucciones PRINT la respuesta está en la misma línea. Siempre que una instrucción PRINT termine con un punto y coma, el siguiente ítem impreso será adyacente al anterior.

PRACTICA 4.3

La función TAB

Introduzca la orden: NEW

Mientras la coma fuerza a la impresión de la siguiente zona y el punto y coma imprime los datos en la siguiente posición de carácter, la función TAB permite a los datos ser impresos en cualquier posición de la línea.

El Apple dispone la pantalla para cuarenta caracteres por línea. Esto es, sólo pueden imprimirse 40 caracteres en una línea. El argumento de la función TAB debe ser una expresión numérica que evalúe un número positivo menor que 256. Siempre debe usarse un punto y coma después de la función TAB.

Introduzca el programa:

```
10 HOME
20 PRINT "1234567890123456789012345678901234567890"
30 PRINT TAB(7);"A"
40 END
```

Introduzca la orden: RUN

¿Cuál es el argumento de la función TAB? _____

¿En qué posición de la pantalla está impresa la cadena A? _____

_____

Ambas respuestas deben ser 7. La función TAB mueve el cursor a la posición especificada, y el dato que sigue a continuación de la función TAB se imprime allí.

Introduzca la sentencia: 30 PRINT TAB(5);"A";TAB(26);"B"

Introduzca la orden: LIST

¿Cuántas funciones TAB hay en la línea 30? _____

_____

_____

Introduzca la orden: RUN

Explique qué hizo el ordenador cuando ejecutó la línea 30 _____

_____

_____

_____

Cuando el ordenador ejecutó la línea 30, la TAB(5) movió el cursor hasta la posición quinta de la línea. El ordenador entonces imprimió la cadena A en esa posición. La siguiente TAB(26) movió el cursor a la posición 26 de la misma línea e imprimió ahí la cadena B.

La función TAB sólo moverá el cursor hacia la derecha, no hacia la izquierda.

Introduzca la sentencia: 30 PRINT TAB(33);"A";TAB(12);"B"

Introduzca la orden: LIST

¿Cuál es el argumento de la primera función TAB? _____

¿Cuál es el argumento de la segunda función TAB? _____

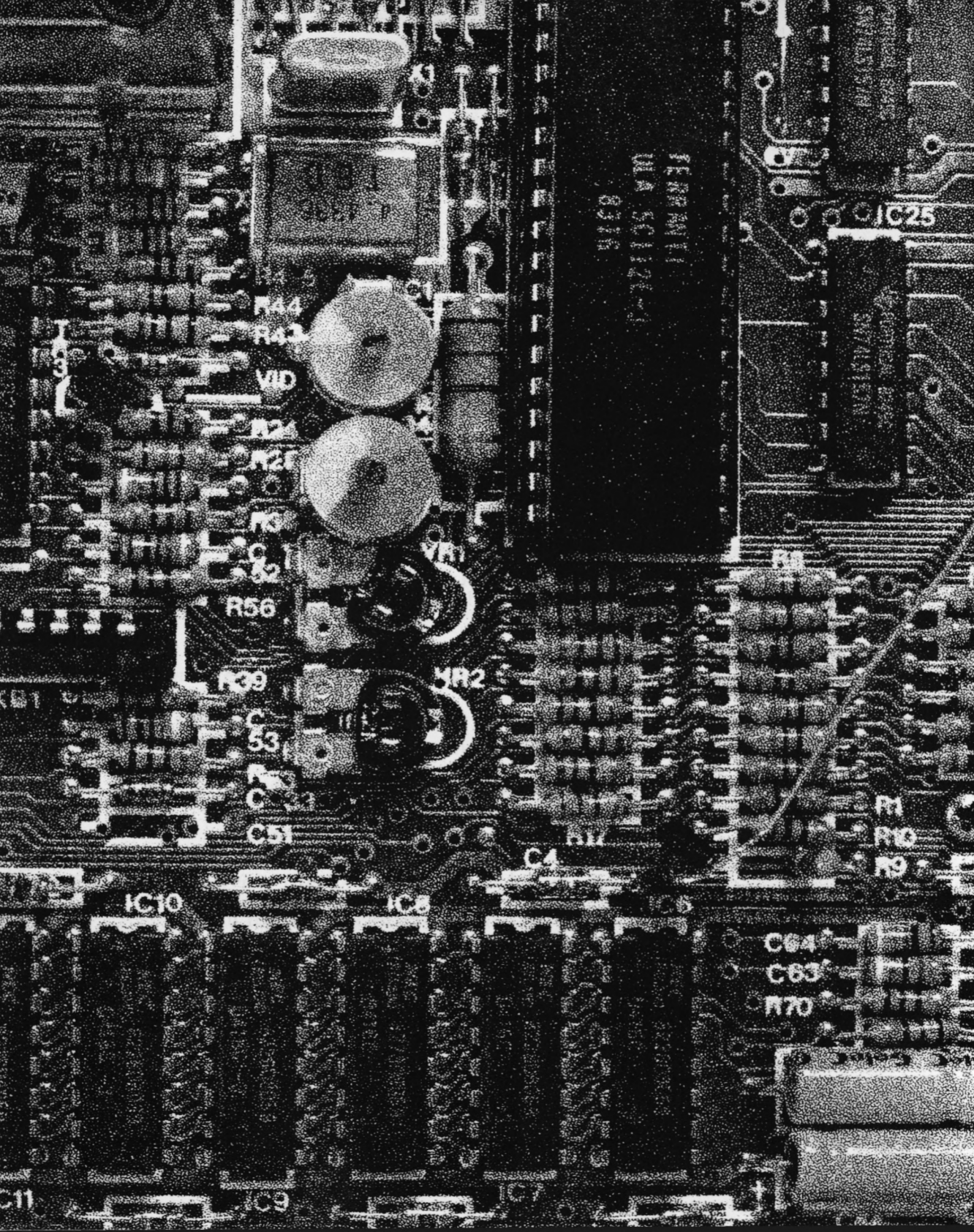
Introduzca la orden: RUN

¿A qué posición se movió el cursor para la primera línea? _____

Cuando se ejecutó TAB(33), el cursor se movió a la posición 33 de la línea.

¿En qué posición está impresa la cadena B? _____

Cuando el ordenador intentó ejecutar TAB(12), el cursor estaba en la posición 34 de la línea. Como la función TAB mueve el cursor sólo hacia la derecha, el cursor no pudo ser movido a la posición 12 de la línea. Así pues, la TAB(12) no fue ejecutada y el ordenador imprimió la cadena en la posición 34.


IC25
R44
VID
R56
R39
C51
C4
R1
R9
IC10
IC8
C64
C63
R70
IC9
IC7

5
Las instrucciones READ/DATA e INPUT

5.0
Introducción

En el capítulo 1 mencionábamos que LET es sólo uno de los cuatro tipos de instrucciones de asignación en el lenguaje BASIC. En este capítulo introducimos dos instrucciones BASIC más para asignar datos a las variables: READ/DATA e INPUT. La cuarta instrucción, GET, es discutida en el capítulo 12.

5.1
La instrucción READ/DATA

Aunque nos vamos a referir a la orden READ/DATA como un todo, realmente son dos instrucciones separadas que trabajan juntas. La instrucción DATA nos preocupa para hacer un listado de los datos que serán asignados a las variables en algún punto del programa. Es otro ejemplo de

instrucción no ejecutable. La asignación real de los valores de una lista de datos se realiza por la instrucción READ. La instrucción READ es seguida de una lista de una o más variables, separados por comas, que están para recibir valores listados en la instrucción DATA.

Ejemplo

```
10 DATA "FRIAS,JOSE", 23,
        HOMBRE
20 READ N$,A,S$
```

N$	A	S$
	0	
FRIAS,JOSE	23	HOMBRE

La idea es simple: el ordenador confronta las variables listadas en la línea 20 con los datos listados en la línea 10. Como usted puede ver, tanto los valores numéricos como los valores tipo cadena pueden ser asignados usando las instrucciones READ y DATA. Los valores de cadena deben ser confrontados con las variables de cadena, y los valores numéricos con las variables numéricas, o de otro modo tendrá lugar un error del tipo TYPE MISMATCH ERROR, y el programa se parará. Las comillas encerrando los datos cadena son opcionales, a menos que la cadena tenga comas, punto y coma o espacios. Para mayor claridad, siempre encerraremos entre comillas todos los datos tipo cadena.

Las comillas pueden usarse como un carácter dentro de una cadena mientras no sean el primer carácter. En una instrucción DATA una cadena que contenga comillas no puede ser encerrada entre comillas.

Ejemplo

```
10 DATA UN ANGULO "AGUDO"
20 READ A$
```

A$
UN "ANGULO" AGUDO

Obsérvese que no hay manera de que la instrucción LET pueda hacer la misma asignación. En la instrucción LET, los datos de cadena deben ir encerrados entre comillas y no pueden contenerlas como un carácter.

Las instrucciones READ deben estar en su programa allí donde se necesita la asignación de datos. Pero las instrucciones DATA pueden estar en cualquier lugar. El ordenador las descubrirá. Para ilustrar esto, en el próximo ejemplo tres programas asignarán 4, 5 y 8 a A, B y C, respectivamente. Las sentencias DATA, sin embargo, aparecen en diferentes partes.

Ejemplo

```
10 DATA 4,5,8
20 READ A,B,C
30 PRINT A*B+C
40 END
```

```
]RUN
 28
```

```
10 READ A,B,C,
20 PRINT A*B+C
30 DATA 4,5,8
40 END
```

```
]RUN
 28
```

```
10 READ A,B,C,
20 PRINT A*B+C
30 END
40 DATA 4,5,8
```

```
]RUN
 28
```

Si hay varias instrucciones DATA en un programa, una buena idea, por lo general, es colocarlas juntas. Así, es más sencillo descubrirlas en caso de que el programa necesite corregirse. El número de instrucciones READ no necesita coincidir con el número de instrucciones DATA en un programa, mientras los ítems de las variables y los datos coincidan. Los dos programas que siguen asignan 5, 6 y 7 a las variables A, B y C, respectivamente.

a)

```
10 READ A, B
20 READ C
30 PRINT A+B+C
40 DATA 5, 6, 7
50 END
```

A	B	C	CRT
0	0	0	
5	6	0	
5	6	7	
5	6	7	(18)

b)

```
10 READ A, B, C
20 PRINT A+B+C
30 DATA 5, 6
40 DATA 7
50 END
```

A	B	C	CRT
0	0	0	
5	6	7	
5	6	7	(18)

El programa *b)* muestra que si hay muy pocos ítems de datos en una instrucción DATA, el ordenador buscará automáticamente en la siguiente instrucción DATA en el programa.

Cuando no hay datos suficientes para satisfacer una instrucción READ, tiene lugar un error del tipo OUT OF DATA ERROR, y el programa se interrumpe.

Ejemplo

```
10 READ A, B, C
20 DATA 5, 6
30 PRINT A+B+C
40 END
```

```
]RUN
?OUT OF DATA ERROR IN 10
]
```

5.1 Ejercicios

1. Realice un diagrama de cajas para cada uno de estos programas:

a)
```
200 READ A,D,F1
210 LET H=2*A-D/2*F1
220 PRINT H
230 DATA 5, 6, -2
240 END
```

b)
```
30 READ R$,T,E,F$
35 PRINT R$,T,E;
40 PRINT F$
50 DATA "TEA"
60 DATA 4,2
70 DATA "?"
80 END
```

c)
```
10 DATA 2, 3, 4
20 READ X, Y, Z
30 LET Z=X+Y+Z
40 PRINT Z
50 END
```

2. Tres de los seis programas siguientes no funcionan. Descúbralos y explique el error que se ha cometido. Para los programas que funcionan, ¿cuál es su respuesta?

a)
```
10 DATA 23, "NO","BIEN",4
20 READ X,C$,T,T,R
25 PRINT X,C$,T,R
30 END
```

b)
```
10 READ X,Y,Z
20 DATA 2,47
25 PRINT X,Y,Z
30 END
```

c)
```
100 READ R$,T
110 DATA "MOZART",4,"MATH"
115 PRINT R$,T
120 END
```

d)
```
40 READ R,T,P
45 PRINT R,T
50 END
```

e)
```
10 READ A,A,A
20 DATA 5,6,7
30 PRINT A,A,A
40 END
```

f)
```
10 DATA 3,5
15 LET X=5
20 LET Y=3
40 READ X,Y
50 PRINT X,Y
60 END
```

3. La aplicación del programa «intereses obtenidos» del capítulo 3 puede ser reescrito usando READ/DATA de la forma siguiente:

```
100 REM INTERESES OBTENIDOS
110 READ P,R
120 LET I=P*R
130 LET P=P+I
140 PRINT "INTERESES OBTENIDOS ";I
150 PRINT "TOTAL DE LA CUENTA DESPUES DE UN MES ";P
160 DATA 5000,.015
170 END
```

Reescriba estos programas del capítulo 3:

a) Balance del talonario de cheques.
b) Solución para un ángulo de un triángulo.
c) Cargo de intereses.
d) Análisis de las existencias de mercado.

Use READ/DATA para asignar los datos a variables.

5.2

Suma reiterativa

La *suma reiterativa* es una suma repetida que usa las mismas variables una y otra vez. Los ejemplos siguientes muestran este proceso. *Importante:* La variable S en el programa siguiente se la conoce como "acumulador". Un acumulador es la tecla "memoria más" (M+) de una calculadora. Conserva un total calculado.

Ejemplo Este programa suma los números 5, 35 y 60.

```
100 REM DEMOSTRACION
105 REM DE SUMA REITERATIVA
110 LET S=0
120 READ X
130 LET S=S+X
140 READ X
150 LET S=S+X
160 READ X
170 LET S=S+X
180 PRINT S
190 DATA 5,35,60
200 END
```

S	X	TRC
0	0	
0	5	
5	5	
5	35	
40	35	
40	60	
100	60	
100	60	(100)

Ciertamente puede obtenerse el mismo resultado con un programa más corto:

```
10 READ X,Y,Z
20 LET S=X+Y+Z
```

```
30 PRINT S
40 DATA 5,35,60
50 END
```

Pero el método reiterativo ofrece ventajas que no serán completamente apreciadas hasta el capítulo próximo. Practíquelo ahora.

Ejemplo La señora Hueso tiene seis hijos. El siguiente programa imprime sus nombres y las edades en la pantalla y calcula su edad media. Emplee la suma reiterativa.

```
100 REM CALCULO DE LA MEDIA DE EDAD
110 LET S=0
120 READ N$,A
130 PRINT N$;" TIENE ";A;"AÑOS."
140 LET S=S+A
150 READ N$,A
160 PRINT N$;" TIENE ";A;"AÑOS."
170 LET S=S+A
180 READ N$,A
190 PRINT N$;" TIENE ";A;"AÑOS."
200 LET S=S+A
210 READ N$,A
220 PRINT N$;" TIENE ";A;"AÑOS."
230 LET S=S+A
240 READ N$,A
250 PRINT N$;" TIENE ";A;"AÑOS."
260 LET S=S+A
270 READ N$,A
280 PRINT N$;" TIENE ";A;"AÑOS."
290 LET S=S+A
300 LET M=S/6
310 PRINT "SU EDAD MEDIA ES ";M
320 DATA "LUIS",6,"CRISTINA",8,"MARIANA",10
330 DATA "IRO",12,"ANA",13,"PABLO",18
340 END
```

5.2

Ejercicios

1. Modifique el programa anterior para calcular la edad media de las hijas de la señora Hueso.

5.3

La instrucción RESTORE

Una vez que cualquiera de los valores de la instrucción DATA ha sido usado, puede ser reutilizado con la instrucción RESTORE.

Ejemplo El siguiente programa será interrumpido por un error OUT OF DATA, ya que hay insuficientes datos para las variables:

```
100 READ A,B,C,D,E,F,G
110 DATA 2,1,5,6,24
120 END
```

Sin embargo, para realmacenar los datos antes de que F y G sean "leídos" por el ordenador haremos lo siguiente:

	A	B	C	D	E	F	G
	0	0	0	0	0	0	0
100 READ A,B,C,D,E	2	1	5	6	24	0	0
110 RESTORE	2	1	5	6	24	0	0
120 READ F,G	2	1	5	6	24	2	1

```
130 DATA 2,1,5,6,24
140 END
```

Fíjese que F y G contienen los dos primeros valores de la lista de datos, ya que *todos* ellos han sido repuestos como resultado de la instrucción RESTORE (algunas versiones del BASIC pueden realmacenar datos selectivamente).

5.3

Ejercicios

1. La siguiente tabla muestra las notas de los exámenes de cuatro estudiantes en la clase de BASIC del señor Suárez:

	Examen 1	Examen 2	Examen 3
Daniel	78	89	56
Blas	56	66	89
Juan	81	55	99
José	66	78	80

Usando la sentencia DATA 78,89,56,56,66,89,81,55,99,66,78,80 escriba un programa que:

a) calcule e imprima en la pantalla la nota media de cada estudiante, y

b) calcule e imprima en la pantalla la nota media de cada examen.

Use la técnica de la suma reiterativa y realmacene los datos entre las dos partes del programa.

5.4

La instrucción INPUT

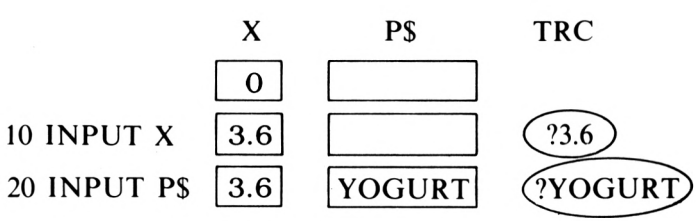
Con la instrucción INPUT, los datos pueden ser asignados a variables directamente desde el teclado por el usuario, a la vez que el programa se está ejecutando. En contraste con las instrucciones LET y READ/DATA que requieren que las asignaciones sean hechas por el programador en el momento en que se escribe el programa. Un programador escribe el programa; un usuario lo ejecuta. Suponga siempre que el usuario no conoce programación y no puede modificar el programa para cambiar los datos en las instrucciones LET o DATA. La instrucción INPUT, en cambio, es una manera de que el usuario pueda proporcionar datos del programa. Los programas que utilizan instrucciones INPUT, se llaman "interactivos".

La forma más simple de la instrucción INPUT es

INPUT X

La variable puede ser numérica o de cadena. Cuando el ordenador encuentra esta instrucción en un programa BASIC, el TRC visualizará una interrogación llamada *indicador* y solicita del usuario una respuesta. Todos los datos suministrados por el usuario en respuesta a una sentencia INPUT se escribirán en este libro subrayadas.

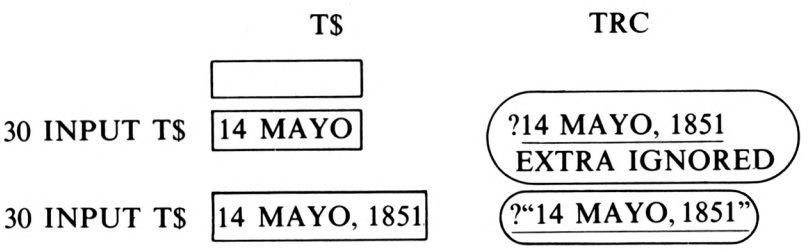
Ejemplo Suponga que deseamos asignar el valor numérico 3.6 a X y la cadena "YOGURT" a P$ utilizando una sentencia INPUT. (Nota: la caja de TRC muestra sólo lo que aparece después de introducir la orden RUN):



No olvide que el subrayado significa que el usuario ha respondido a la indicación INPUT.

No es necesario poner comillas en una cadena que está siendo introducida en respuesta a una instrucción INPUT, a menos que contenga una coma o un punto y coma.

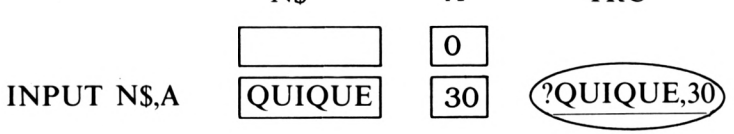
Ejemplo



El mensaje EXTRA IGNORED significa que los datos a la derecha de la coma no fueron asignados a T$.

Pueden asignarse valores a varias variables con una única instrucción INPUT. Esto se hace separando las variables con comas después de la instrucción INPUT.

Ejemplo



En respuesta a la instrucción INPUT, el usuario introduce los datos que desea. Los datos pueden ser escritos todos de una vez separados por comas, o bien puede presionarse RETURN después de cada ítem.

Si el usuario no está familiarizado con el programa que está funcionando, o si hay varias instrucciones INPUT diferentes en el programa, una interrogación indicadora puede no ser suficiente. Para eliminar la confusión, usted (como programador) puede insertar un mensaje indicador entre comillas después de una instrucción INPUT. El BASIC imprimirá este mensaje en vez de la interrogación. Debe ir un punto y coma entre el mensaje indicador y la lista de variables (algunas versiones del BASIC usan coma en vez de punto y coma).

Ejemplo

```
100 INPUT "PONGA SU NOMBRE Y EDAD>>";N$,A
105 PRINT
110 PRINT N$;", HAS EMPLEADO CASI"
120 PRINT A/3;" AÑOS DE TU VIDA DURMIENDO"
130 END
```

```
]RUN
PONGA SU NOMBRE Y EDAD »QUIQUE, 30
QUIQUE, HAS EMPLEADO CASI
10 AÑOS DE TU VIDA DURMIENDO
```

Ejemplo

```
10 REM ESTE PROGRAMA IMPRIME LA SUMA
20 REM DE DOS NUMEROS ARBITRARIOS
30 INPUT "PRIMER NUMERO:";X
40 INPUT "¡EL SIGUIENTE POR FAVOR!";Y
50 PRINT "LA SUMA DE ";X;" Y ";Y;" ES ";X+Y
60 END
```

```
]RUN
PRIMER NUMERO: 3
¡EL SIGUIENTE POR FAVOR! 5
LA SUMA DE 3 Y 5 ES 8
```

APLICACION: ANALISIS DE EXISTENCIAS

El señor Giner compró S1 acciones de IBM por D1 pesetas cada acción y S2 acciones de SONY por D2 pesetas cada acción. El siguiente programa calculará e imprimirá su inversión total. A continuación del programa mostraremos un ejemplo:

```
S1 = 450
D1 = 7550
S2 = 200
D2 = 2525
```

```
100 REM ANALISIS EXISTENCIAS PARA VALORES ARBITRARIOS
110 REM
115 REM    MODULO DE ASIGNACION
117 REM
120 INPUT "CUANTAS ACCIONES DE IBM?";S1
130 INPUT "CUANTO POR ACCION?";D1
140 INPUT "CUANTAS ACCIONES DE SONY?";S2
150 INPUT "CUANTO POR ACCION?";D2
155 REM
160 REM    MODULO DE CALCULO
165 REM
170 LET I=S1*D1+S2*D2
173 REM
175 REM    MODULO DE SALIDA
```

```
178 REM
180 PRINT
185 PRINT "SU INVERSION TOTAL ES ";I;" PESETAS."
190 END
```

```
]RUN
CUANTAS ACCIONES DE IBM? 450
CUANTO POR ACCION? 75,50
CUANTAS ACCIONES DE SONY? 200
CUANTO POR ACCION? 25,25
SU INVERSION TOTAL ES 39025 PESETAS
```

5.4

Ejercicios

1. Suponga que el señor Giner también compró S3 acciones de ITT a D3 dólares cada acción. Modifique el programa anterior para calcular e imprimir en la pantalla el total invertido en cada una de las tres cantidades y el capital total invertido.
2. Modifique el programa llamado «intereses obtenidos» del capítulo 3, sustituyendo las instrucciones LET necesarias por la sentencia INPUT.

5.5

Proyectos

1. APLICACION: LA DESVIACION TIPICA

La desviación típica es utilizada por los estadísticos para medir el alejamiento de los datos respecto de la media estadística. La siguiente tabla mostrará cómo se calcula la desviación típica para los datos listados bajo la X:

X	X − MEDIA	(X − MEDIA)^2
3	0	0
7	4	16
−2	−5	25
8	5	25
3	0	0
0	−3	9
2	−1	1
SUMA = 21		SUMA = 76
MEDIA = 21/7 = 3		MEDIA = 76/7 = 10.857
		DESV. TIP. = SQR(10.857) = 3.295

Si suponemos la sentencia DATA para los valores de X: DATA 3, 7, −2, 8, 3, 0, 2, escriba un programa que calcule e imprima la media y la desviación típica.

2. La señora Caspe compró S1 acciones de IBM a P1 dólares cada una; S2 acciones de SONY a P2 pesetas cada una, y S3 acciones de ITT a P3 pesetas cada una. Varios meses después recibió D1 pesetas por acción como dividendos de IBM; D2 pesetas cada acción como dividendos de SONY y D3 pesetas cada acción como dividendos de ITT. Más tarde vendió las existencias por los siguientes precios:

Q1 pesetas cada acción de IBM
Q2 pesetas cada acción de SONY
Q3 pesetas cada acción de ITT

Escriba un programa BASIC para calcular e imprimir el coste total de compra, el total de dividendos, los ingresos totales recibidos de las ventas y los beneficios netos totales, siendo:

COSTE TOTAL DE COMPRA = SUMA DE (NUMERO DE ACCIONES COMPRADAS * PRECIO POR ACCION)

DIVIDENDOS TOTALES = SUMA DE (NUMERO DE ACCIONES COMPRADAS * DIVIDENDOS POR ACCION)

INGRESO TOTAL RECIBIDO DE LAS VENTAS = SUMA DE (NUMERO DE ACCIONES COMPRADAS * PRECIO DE VENTA POR ACCION)

BENEFICIOS NETOS TOTALES = INGRESO TOTAL POR LAS VENTAS + DIVIDENDOS TOTALES − COSTES TOTALES

En la sección del módulo de asignación del programa asigne todos los valores utilizando solamente instrucciones INPUT.

PRACTICA 5.1

Las instrucciones READ/DATA

Introduzca la orden: NEW

Las instrucciones READ y DATA son utilizadas para asignar datos a variables dentro de un programa. La instrucción DATA es una instrucción no ejecutable que contiene datos. Los datos listados en la instrucción

DATA pueden ser tanto numéricos como cadenas y deben estar separados por comas.

La instrucción READ asigna los datos contenidos en una instrucción DATA. Los argumentos de la instrucción READ son variables y deben estar separados por comas.

Introduzca el programa:
```
10 READ A$,X
20 PRINT A$,X
30 DATA "UNO",2
40 END
```

¿Cuáles son las variables de la instrucción READ? _____

¿Cuáles son los datos de la instrucción DATA? _____

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

Cuando el ordenador ejecuta la línea numerada 10, el primer ítem de la instrucción DATA es asignado a la primera variable de la instrucción READ; el segundo ítem de la instrucción DATA es asignado a la segunda variable de la instrucción READ.

El tipo de variables listadas en la instrucción READ debe coincidir con el tipo de datos de la instrucción DATA.

Introduzca la instrucción: 30 DATA 2,"UNO"

Introduzca la orden: LIST

¿Cuál es el segundo ítem de la instrucción DATA? _____

¿Cuál es la segunda variable de la instrucción READ? _____

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

Introduzca la orden: PRINT A$

¿Cuál es la respuesta del ordenador? _____

El ERROR no fue causado por la asignación de 2 a A$. El ordenador interpretó el dato 2 como una cadena y lo asignó a A$. Sin embargo, el segundo ítem de la instrucción DATA es la cadena UNO, y el intento de asignarlo a la variable numérica X causó el ERROR.

Introduzca la sentencia: 30 DATA "HOLA"

Introduzca la orden: LIST

¿Cuántas variables hay en la instrucción READ? _____

¿Cuántas variables hay en la instrucción DATA? _____

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

El ERROR fue causado por la falta de datos en la instrucción DATA. Debe haber suficientes datos en la instrucción DATA por cada asignación hecha por la instrucción READ.

Introduzca las instrucciones:
```
40 DATA 1,ALLI,2
50 READ A$,X
60 PRINT A$,X
70 END
```

Introduzca la orden: LIST

La instrucción DATA puede estar en cualquier sitio del programa, ya que es una instrucción no ejecutable.

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

Cuando la instrucción READ asigna datos a sus variables, comienza con la instrucción DATA que tenga el número de línea más bajo. Así, fue asignada primero la cadena HOLA. Después cada uno de los datos es asignado por una sentencia DATA distinta, los datos de las siguientes sentencias DATA serán asignados en el orden numérico de los números de línea.

Obsérvese que la cadena ALLI no está entre comillas. Las comillas son necesarias cuando la cadena contiene una coma o dos puntos.

PRACTICA 5.2

RESTORE

Introduzca la orden: NEW

La función de la instrucción RESTORE puede verse mejor en el siguiente programa.

Introduzca el programa:

```
10 READ X
20 RESTORE
30 READ Y
40 RESTORE
50 READ Z
60 PRINT X,Y,Z
70 DATA 4,5,6
80 END
```

Introduzca la orden: RUN

¿Qué valores están asignados a X, Y y Z? _____

El número 4 es asignado a todas las variables. Siempre que se asigna un valor por la sentencia READ, el siguiente valor leído es el ítem siguiente en la sentencia DATA. Sin embargo, cuando se ejecuta la instrucción RESTORE, el valor siguiente que es leído será el primer ítem de la primera sentencia DATA del programa.

Introduzca la orden: NEW

Introduzca el programa:

```
10 READ X
20 READ Y
30 RESTORE
40 PRINT X,Y
50 DATA 3
60 END
```

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

Explique por qué ocurre el error _____

No hay más datos en el número de línea 50 para ser asignados a la variable Y de la línea 20. La sentencia RESTORE de la línea 30 se ejecuta demasiado tarde en el programa para evitar el error.

Introduzca las sentencias:
```
20 RESTORE
30 READ Y
```

Introduzca la orden: LIST

Introduzca la orden: RUN

Obsérvese que la sentencia RESTORE es ejecutada antes que la sentencia READ. Esto permite al primer ítem de la orden DATA ser releído.

PRACTICA 5.3

INPUT

Introduzca la orden: NEW

La instrucción INPUT permite al usuario asignar datos a variables desde el teclado.

Introduzca el programa:
```
10 HOME
20 INPUT X
30 PRINT "EL VALOR DE X ES ";X
40 END
```

El argumento de la instrucción INPUT puede ser una o más variables tipo cadena, o una o más variables numéricas.

¿Cuál es el argumento de la instrucción INPUT de la línea 20?

El argumento de la instrucción INPUT es la variable numérica X.

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador?

Deberemos ver una interrogación (?) en la pantalla. Esta es la manera que tiene el ordenador de pedir una entrada de datos al usuario.

Introduzca el número: 7

¿Cuál es la respuesta del ordenador? _____

El tipo del argumento de la sentencia INPUT y del dato introducido deben coincidir. Esto es, si el argumento de la sentencia INPUT es una variable numérica, sólo serán aceptados números.

Introduzca la orden: RUN

Introduzca la cadena: HOLA

¿Cuál es la respuesta del ordenador? _____

El ordenador hizo como que no se enteró, y escribe el mensaje ?REENTER (reintroduzca) y una interrogación para otra entrada.

Explique por qué el ordenador no acepta la cadena HOLA

_____

_____

El argumento de la instrucción INPUT es una variable numérica. Por tanto, sólo serán aceptados números.

Pulse la tecla RESET.

Pulsando la tecla RESET se interrumpe un programa mientras está funcionando.

Introduzca las sentencias:
```
20 INPUT "PONGA SU NOMBRE: ";X$
30 PRINT "SU NOMBRE ES ";X$
```

Introduzca la orden: LIST

La expresión "PONGA SU NOMBRE:" antes de la variable X$ de la sentencia INPUT se llama *mensaje demandante.* Este debe ser una cadena encerrada entre comillas. La variable y el mensaje demandante de una sentencia INPUT deben estar separados por un punto y coma.

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

La respuesta del ordenador es el mensaje demandante de la sentencia INPUT.

Introduzca la cadena: JUAN PEREZ

¿Cuál es la respuesta del ordenador? _____

Siempre que se usa una variable de cadena en una instrucción INPUT serán aceptadas como entrada todas las cadenas excepto las que tengan comas (,) o dos puntos (:).
Las comas y los dos puntos pueden introducirse si son parte de una cadena que está encerrada entre comillas.

Introduzca la orden: RUN

Introduzca la cadena: PEREZ, JUAN

¿Cuál es la respuesta del ordenador? _____

La respuesta del ordenador, ?EXTRA IGNORED, se produjo por la coma de la cadena PEREZ, JUAN

Introduzca la orden: RUN

Introduzca la cadena: "PEREZ, JUAN"

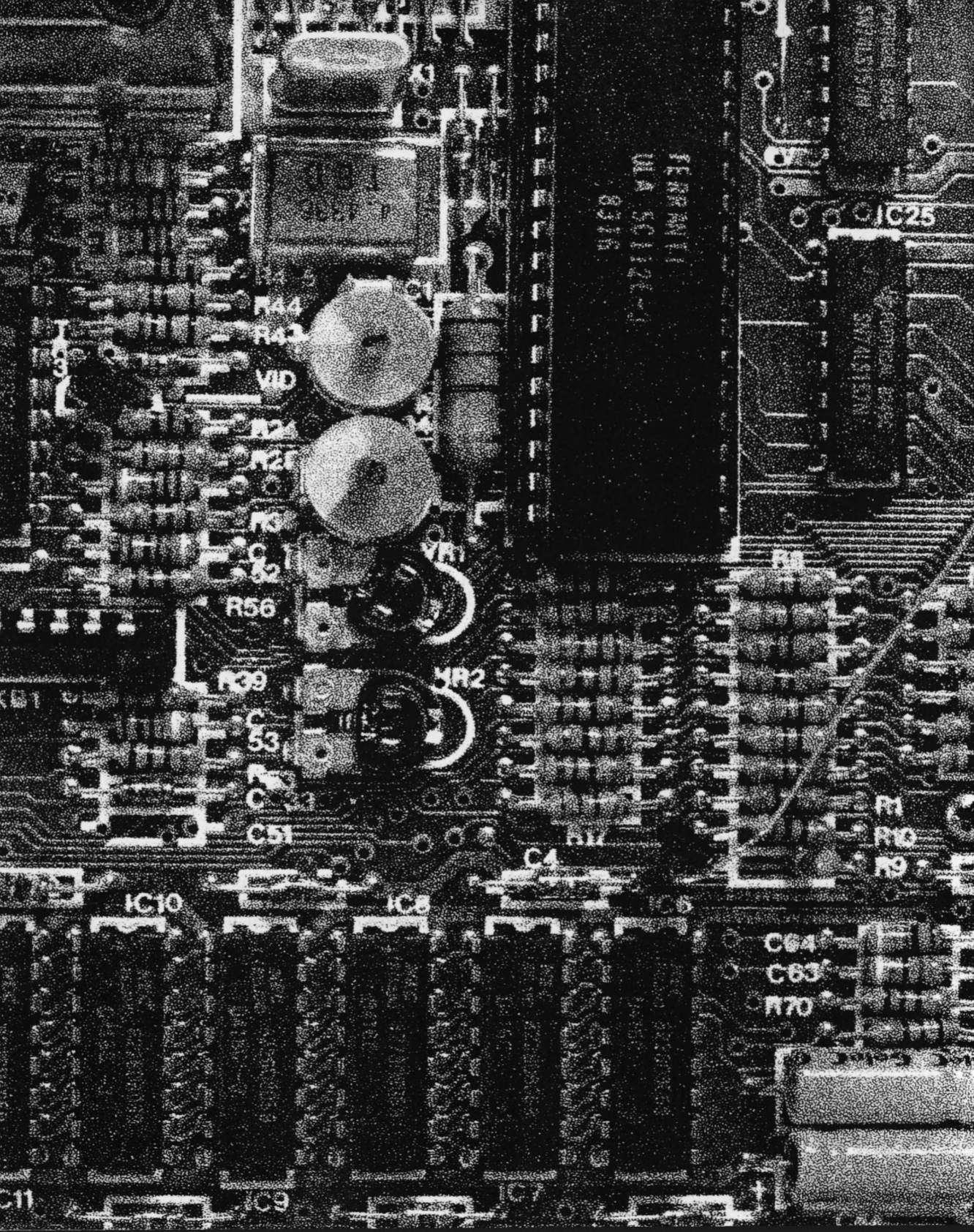
¿Cuál es la respuesta del ordenador? _____

Explique por qué el ordenador aceptó la cadena PEREZ, JUAN.

_____

_____

Recuerde, las comas y los dos puntos pueden introducirse si forman parte de una cadena que está encerrada entre comillas.


IC25
R44
VID
R56
R39
C51
C4
R1
R9
IC10
IC8
C64
C63
R70
IC9
IC7

6
Las instrucciones de decisión y ramificación

6.1
La instrucción IF-THEN

¿Cómo sabe el ordenador decirnos cuándo un vuelo está ya vendido o cuándo tenemos al descubierto nuestra cuenta corriente? ¿Cómo sabe la caja registradora computerizada qué precio asignar a un tarro de salsa? La respuesta nos lleva a la instrucción IF-THEN que dota al ordenador con la habilidad de comprobar una condición y tomar una acción si la condición es cierta. La forma general es

IF *condición* THEN *acción*

(SI... condición, ENTONCES acción)

donde la condición es una expresión que es *verdadera* o *falsa,* y la acción es cualquier instrucción BASIC. La acción es realizada *solamente* si la condición es verdad.

Las normas siguientes gobiernan la instrucción IF-THEN:

a) Si la condición es *falsa,* la instrucción de la acción es ignorada y el programa pasa automáticamente a la próxima línea.

b) Si la condición es *verdad,* es realizada la instrucción de la acción. Después de que la acción es ejecutada, el control pasa a la próxima instrucción BASIC. La próxima instrucción BASIC podrá estar en la misma línea (una línea de instrucciones múltiples) o en la próxima línea (línea de instrucción simple).

Ejemplos de sentencias IF-THEN

```
IF N=250 THEN PRINT "EL VUELO ESTA VENDIDO YA"
IF B-C<0 THEN PRINT "FONDOS INSUFICIENTES"
IF I$="SALSA" THEN LET P=.89
```

Hay seis diferentes formas de comparar datos en una condición. Los símbolos de comparación se llaman *operadores relacionales.*

Operador	*Significado*
=	es igual a
<	es menor que
>	es mayor que
<=	es menor o igual que
>=	es mayor o igual que
<>	no es igual a

Ejemplo	A	B	X	CTR
10 LET A=2 : LET B=4	2	4	0	
20 LET X=10	2	4	10	
30 IF A=B-2 THEN PRINT "IGUAL"	2	4	10	IGUAL
40 IF X<B+8 THEN LET A=1	1	4	10	
50 IF A<>2 THEN LET X=5	1	4	5	
60 IF "HOLA"="MIO" THEN LET A=2	1	4	5	
70 IF "HOLA"<>"MIO" THEN LET A=0	0	4	5	

La relación B>18 comprueba la condición sobre un número y una variable numérica. Pero, ¿podría intentar comparar cadenas? ¿Qué quiere decir realmente R$>"CRISTAL"?

La respuesta es que una cadena es considerada "más grande" que otra

si alfabéticamente viene después. Imaginemos que estamos buscando ambas cadenas en un diccionario. La que encontremos más cercana al principio se dice que es "menor que" la otra. Suponga que R$ contiene la cadena UVAS. La relación R$ > "CRISTAL" es *verdad*. Una relación puede comprobar una condición en valores numéricos o en valores de cadena, pero no puede haber mezcla de los dos tipos de datos.

No se puede preguntar si la cadena SOPA es mayor que 14.

Precaución: En el Applesoft, ninguna comparación con una sentencia IF-THEN puede terminar con "A", como en

```
IF B<A THEN LET C=5
```

Esto es porque "AT" es una palabra reservada. Y la sentencia, cuando sea listada, aparecería así:

```
IF B<AT HEN LET C=5
```

Para evitar este problema, use paréntesis:

```
IF (B<A) THEN LET C=5
```

6.1 Ejercicios

1. Diga si las condiciones son verdaderas o falsas.

   a) A$ > "HOGAR" cuando A$ es CASA
   b) Z$ < A$ donde Z$ es CHIMPANCE y A$ es CHIMPANCES
   c) C$ > = D$ donde C$ y D$ son ambos PURPURA
   d) F$ < "M" donde F$ es "MAXWELL"

2. Explique qué está equivocado en las sentencias IF-THEN:

   a) IF X$ < > 2 THEN LET F = 3
   b) IF B = "4" THEN PRINT R + 2*B
   c) IF PRINT R THEN LET X = 4
   d) IF B < > 3 THEN X4$ < > "HOLA"

3. Realice un diagrama de cajas para el siguiente programa:

```
100 LET X=2
110 LET X1$="HOLA"
120 LET T=4
130 LET Z$="ADIOS"
```

```
140 IF X+1<>2 THEN LET T=0
150 IF T=0 THEN PRINT X
160 IF X1$<Z$ THEN LET Z$="MAÑANA"
170 IF T>X+6 THEN LET T=5
180 END
```

CONECTIVAS LOGICAS

Las condiciones pueden ser conectadas por dos *conectivas* AND y OR.

Las siguientes normas se aplican para conectar condiciones:

a) La expresión

*condición*1 AND *condición*2
es *verdad* sólo si la condición 1 y la condición 2 son ambas *verdad*. De otra manera es *falsa*.

b) La expresión

*condición*1 OR *condición*2
es *verdad* cuando al menos una de las condiciones es *verdad*. De otra manera es *falsa*.

Ejemplos de conectivas lógicas

Para los ejemplos siguientes suponga X=2, A=3, B=5. A$="NO".

	X	A	B	A$
	2	3	5	NO
IF X>2 AND A=3 THEN LET B=4	2	3	5	NO
IF A$<>"SI" OR B>=100 THEN LET X=6	6	3	5	NO
IF B<=5 AND A$="NO" THEN LET A$="SI"	6	3	5	SI
IF X=1 OR (B<>3 AND X=B+1) THEN LET B=8	6	3	8	SI

Los ejemplos descubren dos aspectos más. Tanto las relaciones numéricas como las de cadena pueden aparecer juntas en la misma expresión lógica. También se pueden usar los paréntesis para identificar qué parte de la expresión lógica ha de ser evaluada primero.

Los defectos de impresión son originales y no están relacionados con el escaneo.

...son introducidos para

6.1

Ejercicios (cont.)

4. Realice un diagrama de cajas para los siguientes programas:

a)

```
100 LET R=2
110 LET V1=45
120 LET K$="EL"
130 IF K$>"ELLA" OR V1=100 THEN LET R=R+1
140 IF K$<"ELLA" OR V1=100 THEN LET R=R-2
150 IF K$<"ELLA" AND V1 <> 100 THEN LET R=R-2
160 END
```

b)

```
10 LET G3$="GATO"
20 LET T$="PERRO"
30 LET R=3
40 IF G$<>"KITTY" AND R<2 THEN LET Y=Y+R*2
50 IF G3$ <>"KITTY" OR R <2 THEN LET Y=Y+R*2
60 IF (R=5 AND T$="PERRO") OR T$<>"MESTIZO" THEN
   LET T$="CHINO"
70 IF (R=5 OR T$="CACHORRO") AND T$<>"MESTIZO"
   THEN LET T$="HUESO"
```

LINEAS DE INSTRUCCIONES MULTIPLES

En el Applesoft pueden seguir a THEN varias instrucciones, siempre y cuando estén separadas por dos puntos (:).

Ejemplo Líneas de instrucciones simples:

```
10 IF B-C<0 THEN PRINT "FONDOS INSUFICIENTES"
20 IF B-C<0 THEN LET B1=ABS(B-C)
30 IF B-C<0 THEN PRINT "USTED ESTA AL DESCUBIERTO
     POR VALOR DE " ;B1;"PESETAS"
```

Líneas de instrucciones múltiples:

```
10 IF B-C<0 THEN PRINT "FONDOS INSUFICIENTES" : LET
     B1=ABS(B-C) : PRINT "USTED ESTA AL DESCUBIERTO
     POR VALOR DE " ;B;"PESETAS"
```

Los defectos de impresión son originales y no están relacionados con el escaneo.

.1 " FUNCIONA!"

. - 1. Así que la respuesta es

Ejemplo

```
100 LET X=1 : IF X>1 THEN L ET Y=5: LET Z=4
110 LET Y=44
```

X	Y	Z
1	0	0
1	44	0

En este caso la X no es mayor que 1. Por eso no es ejecutada *ninguna* de las instrucciones que siguen a THEN en la línea 100. La ejecución continúa en la línea 110.

Ejemplo

```
10 LET L$="2" : IF 2=VAL(L$) THEN LET X =1 :
   LET Z=4 : LET Y=9
20 LET L$="TRES"
```

X	Y	L$	Z
1	9	2	4
1	9	TRES	4

En este caso VAL(L$) es igual a 2. De ahí que las sentencias que siguen a THEN en la línea 10 sean ejecutadas.

Los dos puntos pueden usarse para unir tantas sentencias como se necesite en una línea de programa. Una línea de programa puede tener hasta 239 caracteres, que es casi seis líneas de la pantalla en un Apple de cuarenta caracteres por línea.

Precaución: Dada la sentencia

```
100 REM SUMA DE DOS NUMEROS : LET=Y+Z
```

el ordenador no ejecutará la sentencia LET que está a la derecha de los dos puntos, porque es considerada como parte de la anotación REM.

6.1

Ejercicios (cont.)

5. Este programa descubre el mayor de dos números:

```
 5 REM DESCUBRE EL MAYOR DE DOS NUMEROS
10 INPUT X,Y
20 LET L=X
30 IF Y>L THEN LET L=Y
40 PRINT "EL MAYOR ES ";L
50 END
```

a) Realice un diagrama de cajas cuando son introducidos para X e Y los valores 35 y 72.

b) Modifique el programa de tal manera que sea impreso en la pantalla el menor de los dos números.

c) Modifique el programa de tal manera que imprima un mensaje de si los dos números son iguales, o imprima el mayor si no lo son.

6. El siguiente programa BASIC descubre el mayor de tres números.

```
100 REM HALLA EL MAYOR DE TRES NUMEROS
120 INPUT X,Y,Z
130 LET L=X
140 IF L<Y THEN LET L=Y
150 IF L<Z THEN LET L=Z
160 PRINT "EL MAYOR ES ";L
170 END
```

```
]RUN
?5 , 23 , 17
EL MAYOR ES 23
```

a) Realice un diagrama de cajas para los valores 5, 23 y 17.

b) Explique el propósito de la variable L.

c) Modifique el programa para hallar el menor de cuatro números.

7. La tienda Nutty Nut vende cierta variedad de frutos secos: cacahuetes, nueces y anacardos. Los precios de las tres clases de frutos son:

185 pesetas el Kg de cacahuetes,
234 pesetas el Kg de nueces, y
485 pesetas el Kg de anarcardos.

El siguiente programa BASIC calculará el coste total pormenorizado de una venta a un cliente:

```
100 REM PROGRAMA PARA CALCULAR EL PRECIO TOTAL
    PORMENORIZADO DE VENTAS DE FRUTOS SECOS.
110 INPUT "QUE CLASE DE FRUTO QUIERE?";N$
120 IF N$="CACAHUETES" OR N$="NUECES" OR
    N$="ANACARDOS" THEN LET X=1
130 IF X=0 THEN PRINT "LO SIENTO, NO NOS QUEDAN";N$:
    GOTO 200
140 PRINT "CUANTOS KILOS QUIERE USTED DE ";N$;
```

```
150 INPUT P
160 IF N$ = "CACAHUETES" THEN LET Q = P*185
170 IF N$ = "NUECES" THEN LET Q = *P234
180 IF N$ = "ANACARDOS" THEN LET Q = P*485
190 PRINT "EL PRECIO QUE DEBE PAGAR ES ";Q
200 END
```

```
]RUN
QUE CLASE DE FRUTO QUIERE? CACAHUETES
CUANTOS KG QUIERE DE CACAHUETES? 3
EL PRECIO QUE DEBE PAGAR ES 555
```

a) En la línea 120, X se llama "señal" (*flag*, en inglés). Explique su función en este programa.

b) Modifique el programa de tal manera que si un cliente desea comprar varias de las clases de frutos secos, el programa imprima el coste total.

c) Reescriba el programa para utilizar la sentencia DATA:

DATA 185, 234, 485

b) ¿Cuál es el propósito del punto y coma al final de la línea 140?

e) ¿Cómo puede este programa ser reescrito para evitar las conjunciones lógicas en la línea 120?

6.2

La sentencia GOTO

En el capítulo 3 hemos visto que las instrucciones de un programa BASIC son normalmente realizadas en orden creciente. Es posible, sin embargo, cambiar esta secuencia normal. Alterar el flujo normal de un programa se llama "ramificación". Conocida como una "ramificación incondicional", la sentencia GOTO origina un salto que cambia el flujo de un programa.

Ejemplo

```
10 REM ESTE PROGRAMA IMPRIME RAICES CUADRADAS
20 INPUT "INTRODUZCA UN NUMERO POSITIVO ";X
30 PRINT SQR(X)
40 GOTO 20
50 END
```

```
]RUN

INTRODUZCA UN NUMERO POSITIVO 2
1.41421356
INTRODUZCA UN NUMERO POSITIVO 53
7.28010989
INTRODUZCA UN NUMERO POSITIVO (Pulse CTRL-C y RETURN)
]BREAK IN 20
```

La línea 40 envía el control de vuelta a la línea 20; así, las líneas 20, 30 y 40 son ejecutadas una y otra vez. Presionando CTRL-C y después RETURN se rompe este bucle infinito. Aparece un mensaje, que nos dice la línea numerada que estaba procesándose en el momento de la interrupción (presionando RESET también podría romperse el lazo infinito, pero no aparecerá el mensaje).

La sentencia GOTO es muy poderosa cuando se usa como la acción de una sentencia IF-THEN. La sentencia IF-THEN GOTO es conocida como una instrucción de "ramificación o salto condicional". Se ejecuta el salto sólo cuando la condición IF es *verdad.*

Ejemplo Reescribamos el programa de ejemplo anterior:

```
10 REM ESTE PROGRAMA IMPRIME RAICES CUADRADAS
20 INPUT "INTRODUZCA UN NUMERO POSITIVO ";X
30 PRINT SQR(X)
40 INPUT "QUIERE PAGAR? ";D$
50 IF D$ < > "SI" THEN GOTO 20
55 PRINT "¡ADIOS!"
```

```
]RUN

INTRODUZCA UN NUMERO POSITIVO 2
1.41421356
QUIERE PARAR? NO
INTRODUZCA UN NUMERO POSITIVO 53
QUIERE PARAR? SI
¡ADIOS!
```

6.2 Ejercicios

1. Si reemplazáramos la línea 50 anterior por

   50 IF D$ = "NO" THEN GOTO 20

   ¿qué diferencia habría?

2. ¿Por qué cree usted que sería preferible comparar D$ a SI, como hicimos en el ejemplo, que compararla con NO como en el problema 1? (piense sobre ello desde el punto de vista de un usuario).

Los siguientes ejemplos muestran unos pocos de los muchos usos de la sentencia IF-THEN GOTO.

Ejemplo El siguiente programa sumará la lista de números 34, 66, 89, 53, −23, 134, 45:

```
100 REM SUMA DE UNA LISTA DE NUMEROS
110 DATA 34,66,89,53,-23,134,45,0
120 LET S=0
130 READ X
150 LET S=S+X: REM SUMA REITERATIVA
160 IF X<>0 THEN GOTO 130
180 PRINT "LA SUMA ES ";S
```

```
]RUN
LA SUMA ES 398
```

Nótese que el último valor, 0, es utilizado como una señal que indica el final de la lista de datos.

Ejemplo El siguiente programa BASIC sumará todos los números pares de 2 a 100.

```
100 REM SUMA DE TODOS LOS NUMEROS PARES DE 2 A 100
112 REM
113 REM    MODULO DE ASIGNACION
115 LET S=0
120 LET D=2
125 REM
128 REM    MODULO DE CALCULO
130 LET S=S+D
140 IF D=100 THEN GOTO 160
145 LET D=D+2
150 GOTO 130
155 REM
158 REM    MODULO DE SALIDA
160 PRINT "LA SUMA ES ";S
170 END
```

```
]RUN
LA SUMA ES 2550
```

Ejemplo Las siguientes tres líneas de programa también calculan e imprimen la suma de los números pares desde 2 a 100:

```
10 LET I=0 : LET S=0
20 LET I=I+2 : IF I<=100 THEN LET S=S+I : GOTO 20
30 PRINT "LA SUMA ES ";S :END
```

Preferimos el programa estructurado y documentado del ejemplo anterior. No sacrifique claridad por brevedad en sus programas.

6.2

Ejercicios (cont.)

3. Escriba un programa para sumar todos los números impares de 3 a 1001.

4. Halle la suma de los 50 primeros términos de la serie:

   $1 + (1/2)^2 + (1/3)^2 + (1/4)^2 + ...$

Ejemplo El señor Pérez usa su ordenador personal para apuntar números de teléfono importantes. Suponga que su lista de números de teléfono es:

TIENDA DE MANTEQUERIA DE JOSE	234-5678
GUILLERMO GONZALEZ	435-6789
MARI LUZ SANZ	234-8888
PILAR GOMEZ	345-5567
BANCO CANTABRICO	445-5678

El siguiente programa permitirá al señor Pérez encontrar un número de teléfono introduciendo sólo el nombre:

```
10 REM DIRECTORIO DE TELEFONOS
20 DATA "TIENDA DE MANTEQUERIA DE JOSE","234-5678"
25 DATA "GUILLERMO GONZALEZ", "435-6789"
30 DATA "MARI LUZ SANZ","234-8888"
35 DATA "PILAR GOMEZ","345-5567"
40 DATA "BANCO CANTABRICO","445-5678"
45 DATA "*","*"
50 INPUT "ESCRIBA EL NOMBRE, POR FAVOR ";N$
60 READ X$,Y$
70 IF X$="*" THEN PRINT "NO HAY TAL NOMBRE" : GOTO 110
```

```
 90 IF X$ < > N$ THEN GOTO 60
100 PRINT N$;"SU NUMERO DE TELEFONO ES ";Y$
110 END
```

```
]RUN
ESCRIBA EL NOMBRE POR FAVOR PILAR GOMEZ
PILAR GOMEZ, SU NUMERO DE TELEFONO ES 345-5567
```

6.2

Ejercicios (cont.)

5. El asterisco (*) se usó como llamada para señalar el final de la lista de datos. ¿Por qué hay dos?
6. Escriba un programa BASIC que sea de dirección inversa. El usuario debe introducir el número de teléfono y el nombre aparecerá en la pantalla. Use las mismas sentencias DATA.
7. Modifique el programa de tal manera que también pueda contener domicilios.

Ejemplo El siguiente programa es una modificación del programa de «balance del talonario de cheques» del capítulo 3. Esta versión permite múltiples entradas de depósitos y cheques:

```
100 REM BALANCE DEL TALONARIO DE CHEQUES
105 REM MODULO DE ASIGNACION
110 INPUT "INTRODUZCA EL SALDO ACTUAL: ";B
120 INPUT "¿DESEA HACER UN DEPOSITO? ";A$
125 LET D=0
130 IF A$="SI" THEN INPUT "HAGA EL DEPOSITO: ";D
140 INPUT "INTRODUZCA EL VALOR DEL CHEQUE: ";C
145 REM MODULO DE CALCULO
150 LET B=B+D-C
170 INPUT "¿DESEA HACER OTRA ENTRADA? ";A$
190 IF A$="SI" THEN GOTO 120
195 REM MODULO DE SALIDA
200 PRINT "SU NUEVO SALDO ES ";B
210 END
```

```
]RUN
INTRODUZCA EL SALDO ACTUAL: 5900
DESEA HACER UN DEPOSITO? SI
HAGA EL DEPOSITO: 20000
INTRODUZCA EL VALOR DEL CHEQUE: 2525
DESEA HACER OTRA ENTRADA? NO
SU NUEVO SALDO ES: 23375
```

Ejercicios (cont.)

8. Explique por qué es necesaria la línea 125.

9. Modifique el programa de tal manera que todos los cheques puedan ser introducidos al mismo tiempo. Use la suma reiterativa.

6.3

Proyectos

1. Escriba un programa BASIC que halle el mayor número en una lista arbitraria de números. Permita que los números puedan ser introducidos por el usuario.

2. Escriba un programa BASIC que

   1) permita al usuario introducir tres palabras arbitrariamente.
   2) ordene alfabéticamente las palabras en orden ascendente.
   3) imprima la lista ordenada.

3. APLICACION: CALCULO DEL IMPUESTO SOBRE LA RENTA

   En el país de Applelandia, cuya moneda en curso es el applex, las tablas del Impuesto Sobre la Renta son las siguientes:

   TABLA A
   ESTADO CIVIL: SOLTERO

AJUSTE DE RENTA	IMPUESTO A PAGAR
0 - 5000	5% de la renta ajustada
5001 - 10000	250 + 10% del excedente a 5000
10001 - 20000	1250 + 20% del excedente a 10000
20001 o más	5250 + 40% del excedente a 20000

   TABLA B
   ESTADO CIVIL: CASADO

AJUSTE DE RENTA	IMPUESTO A PAGAR
0 - 5000	2% de la renta ajustada
5001 - 10000	100 + 7% del excedente a 5000
10001 - 20000	800 + 15% del excedente a 10000
20001 y más	3800 + 32% del excedente a 20000

El gobierno permite un descuento de 800 applexs por cada persona dependiente.

Escriba un programa sobre la renta de un individuo bajo las condiciones siguientes:

I. MODULO DE ASIGNACION
Use sólo sentencias INPUT para asignar lo siguiente:
1) Estado civil.
2) Número de personas dependientes.
3) Renta.

II. MODULO DE CALCULO
Calcule la renta ajustada y el impuesto sobre la renta.

III. MODULO DE SALIDA
Imprimir el impuesto sobre la renta.

He aquí una explicación y un ejemplo:

La fórmula para la *renta ajustada* es

RENTA AJUSTADA = RENTA − (# de dependientes) * 800 applexs.

Suponga que la señora Carter tiene una renta total anual de 28500 y declara tres personas dependientes a su cargo. Su renta ajustada será

28500 − 3 * 800 = 26100

Está casada, por lo que aplicando la tabla B, tenemos que su renta ajustada entre en la categoría de "20001 applexs o más". Su impuesto será 3800 + 32 % de 6100, esto es, 5752 applexs.

4. APLICACION: LISTA DE BIENES INMUEBLES

La compañía OK REALTY CO. tiene cinco nuevas casas en lista:

DOMICILIO	# DORMITORIOS	# BAÑOS	PRECIO
Pradillo, 25	5	3	99995
Santa Clara, 248	3	2	125900
Galicia, 48	6	2	235400
Pelícano, 111	5	4	125000
Arenas, 5	3	1	67500

Escriba los siguientes programas:

1) Que se pueda introducir una escala de precios y que el programa encuentre las casas que entran dentro de la escala.

2) Que puedan ser introducidas escalas de precios y del número de habitaciones, y que el programa pueda hallar las casas que satisfacen tales condiciones.

Use las sentencias DATA para almacenar la información en la tabla y use sentencias INPUT para introducir las escalas de precios y de habitaciones.

PRACTICA 6.1

La instrucción IF-THEN

Introduzca la orden: NEW

A la IF (comparación) THEN (instrucción) se la llama sentencia condicional o de decisión.

La instrucción que sigue a THEN se ejecuta sólo cuando la comparación que sigue a IF es verdad.

Introduzca el programa:
```
10 LET X=5
20 IF X=5 THEN PRINT "X ES CINCO"
30 END
```

Introduzca: RUN

¿Cuál es la respuesta del ordenador? _____

Cuando el ordenador ejecuta la línea número 20, primero determina si la comparación (X=5) es verdadera o falsa. Como la comparación es cierta, el ordenador ejecuta la instrucción (PRINT "X ES CINCO") que sigue a THEN.

Introduzca las sentencias:
```
30 PRINT "NO ES IGUAL A CINCO"
40 END
```

Introduzca la orden: LIST

¿Cuál es la respuesta del ordenador? _____

_____

Ya que la comparación X=5 es VERDAD en la línea 20, la instrucción PRINT "X ES CINCO" es ejecutada. Después de ejecutar esta instrucción el ordenador ejecuta la siguiente línea numerada, la 30.

Introduzca la sentencia: 20 IF X<>5 THEN PRINT X

Introduzca la orden: LIST

¿Será cierta la comparación de la línea 20? _____

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

Cuando el ordenador ejecuta la línea número 20, determina si la comparación (X<>5) es verdadera o falsa. Como la comparación es falsa, el ordenador ignora la instrucción que sigue a THEN y ejecuta la siguiente línea numerada.

Las conjunciones lógicas (AND, OR) pueden usarse en la comparación.

Introduzca las sentencias: 20 IF X=5 OR X=7 THEN PRINT 20
30 IF X=5 AND X=7 THEN PRINT 30

Introduzca la orden : LIST

La línea numerada 20 usa la conjunción OR en su comparación, y la línea 30 usa la conjunción AND en su comparación.

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

Siempre que la comparación use la conjunción OR, sólo uno de los términos necesita ser verdad para que la comparación completa lo sea también. Sin embargo, cuando se utiliza la conjunción AND, ambos términos necesitan ser ciertos para que la comparación entera sea verdad.

PRACTICA 6.2

Ramificación

Introduzca la orden: NEW

Hay dos tipos de ramificaciones o saltos, el condicional y el incondicional.

La instrucción GOTO se llama salto incondicional. Cuando el ordenador ejecuta esta instrucción, el programa salta a la línea numerada de la instrucción GOTO.

Introduzca el programa:
```
10 GOTO 30
20 PRINT "LINEA NUMERADA 20"
30 PRINT "LINEA NUMERADA 30"
40 END
```

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

Fíjese que no fue ejecutada la línea 20. Cuando el ordenador ejecutó la línea 10, el programa transfirió el control a la línea 30.

El programa siguiente es un ejemplo de un lazo infinito.

Introduzca el programa:
```
10 LET X=0
20 LET X=X+1
30 PRINT X
40 GOTO 20
50 END
```

Introduzca la orden: RUN

Cuando vea demasiados números en la pantalla, presione la tecla RESET.

La ramificación condicional usa la sentencia IF-THEN junto a la GOTO.

Introduzca la sentencia: 40 IF X<=10 THEN GOTO 20

Introduzca la orden: LIST

La línea 40 es una ramificación condicional. El salto ocurrirá sólo cuando la condición, X<=10 sea verdad.

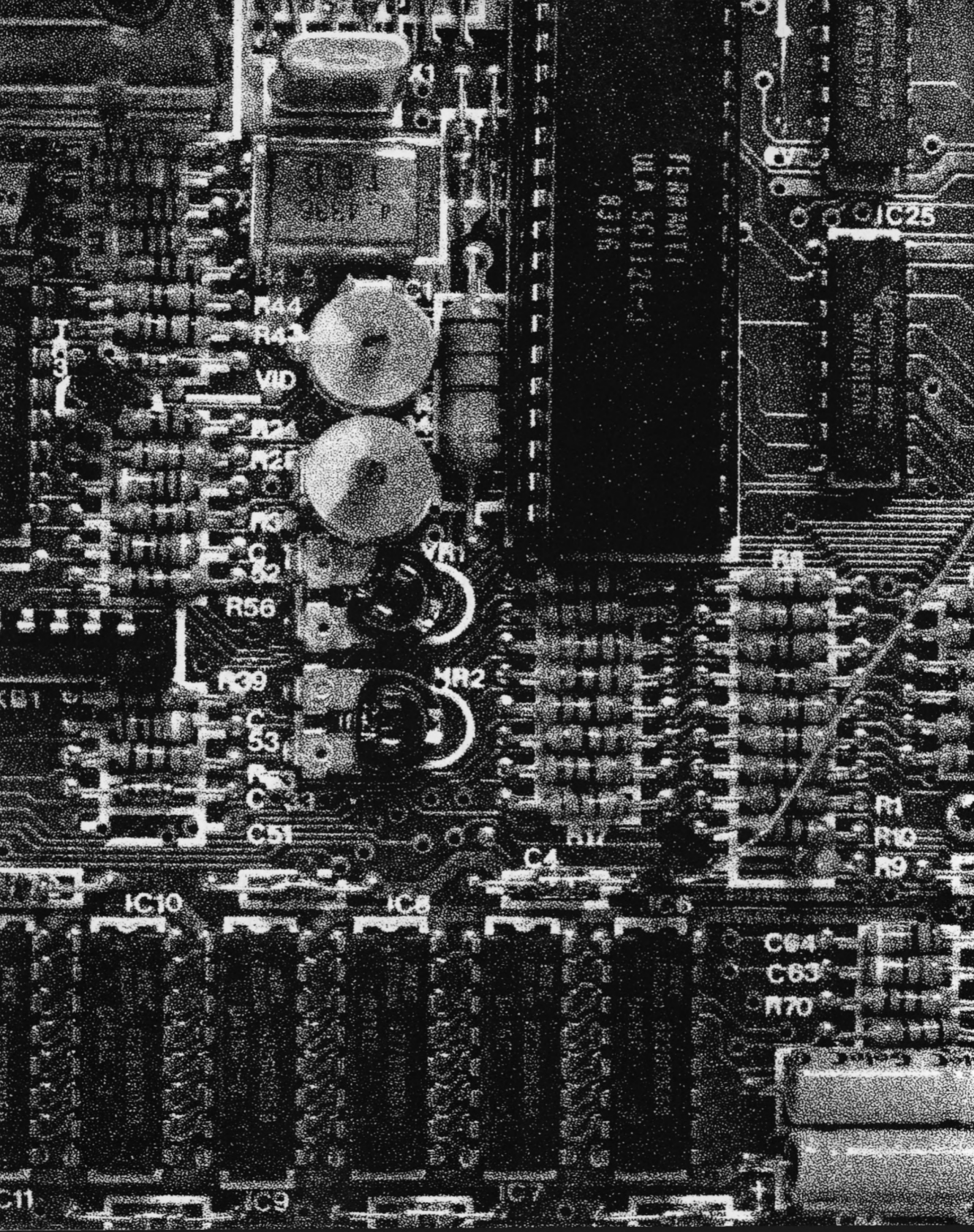
Introduzca la orden: RUN

Esta vez el programa llega hasta el final. Explique el porqué.

Introduzca la orden: PRINT X

¿Cuál es la salida del ordenador? _____

El valor de X es 11. La condición de la línea 40 es falsa. El control, por consiguiente, pasa a la línea 50, la cual termina el programa.


IC25
R44
VID
R56
R39
C51
C4
R1
R9
IC10
IC8
C64
C63
R70
IC9
IC7

7

La instrucción FOR-NEXT

7.0

Introducción

En el capítulo 6 vimos cómo las sentencias IF-THEN y GOTO pueden usarse para repetir una sección del programa una y otra vez. La utilización de la ramificación para este propósito se llama *enlazado (looping)*. El siguiente programa BASIC usa un bucle para sumar los números pares de 2 a 100:

```
100 REM SUMA DE LOS NUMEROS PARES DESDE 2 A 100
115 LET S=0
120 LET K=2
130 LET S=S+K
140 IF K=100 THEN GOTO 160
150 LET K=K+2
155 GOTO 130
160 PRINT S
170 END
```

```
]RUN
2550
```

El enlazado tiene tantas aplicaciones importantes en programación que, por ejemplo, en el lenguaje BASIC se han incluido un grupo de instrucciones dedicadas a la creación de bucles. Estas son las instrucciones FOR y NEXT, que simplifican enormemente la construcción de bucles. Ya que ambas deben funcionar juntas, nos referiremos a ellas como la sentencia FOR-NEXT.

7.1

Comprensión de la instrucción FOR-NEXT

Hay dos formas de sentencia FOR-NEXT:

```
(1) FOR K = M TO N
    .........
    .........
    .........
    NEXT K

(2) FOR K = M TO N STEP Q
    .........
    .........
    .........
    NEXT K
```

Siendo .....cualquier instrucción BASIC, y M, N y Q representan números, variables numéricas, o expresiones aritméticas, y a K se le llama índice o contador. Cualquier variable numérica puede ser usada como contador.

FORMA 1

La forma (1) de la sentencia FOR-NEXT opera bajo las siguientes normas:

1. La instrucción FOR K = M TO N da a K el valor M y se ejecutan las instrucciones BASIC hasta la sentencia NEXT K.
2. La instrucción NEXT K aumenta el valor de K en uno, se detiene a ver si K es *mayor* que N, y si no, vuelve el control a la instrucción que sigue a la sentencia FOR.
3. Cuando la instrucción NEXT K aumenta K a un valor mayor que N, el programa irá a la primera instrucción de después de la sentencia NEXT K.

Ejemplo Este programa hace tres veces una suma reiterativa de la variable S:

```
 5 LET S=0
10 FOR K=2 TO 4
20 LET S=S+K
30 NEXT K
40 PRINT S
50 END
```

```
]RUN
9
```

La instrucción FOR-NEXT repite el lazo, o bucle, tres veces. A continuación mostramos el programa paso a paso con su diagrama de cajas.

	K	S	TRC
5 LET S=0	[0]	[0]	
10 FOR K=2 TO 4	[2]	[0]	
PRIMER LAZO			
20 LET S=S+K	[2]	[2]	
30 NEXT K	[3]	[2]	
SEGUNDO LAZO			
20 LET S=S+K	[3]	[5]	
30 NEXT K	[4]	[5]	
TERCER LAZO			
20 LET S=S+K	[4]	[9]	
30 NEXT K	[5]	[9]	
40 PRINT S	[5]	[9]	(9)
50 END	[5]	[9]	

Ejemplo

```
100 FOR I=-1 TO 2
110 PRINT I+2
120 NEXT I
130 IF I=3 THEN PRINT "ENTENDIDO?"
140 END
```

```
]RUN
1
2
3
4
ENTENDIDO?
```

	I	TRC
100 FOR I = −1 TO 2	[-1]	
PRIMER LAZO		
110 PRINT I+2	[-1]	(1)
120 NEXT I	[0]	
SEGUNDO LAZO		
110 PRINT I+2	[0]	(2)
120 NEXT I	[1]	
TERCER LAZO		
110 PRINT I+2	[1]	(3)
120 NEXT I	[2]	
CUARTO LAZO		
110 PRINT I+2	[2]	(4)
120 NEXT I	[3]	
130 IF I=3 THEN PRINT "¿ENTENDIDO?"	[3]	(¿ENTENDIDO?)
140 END	[3]	

7.1

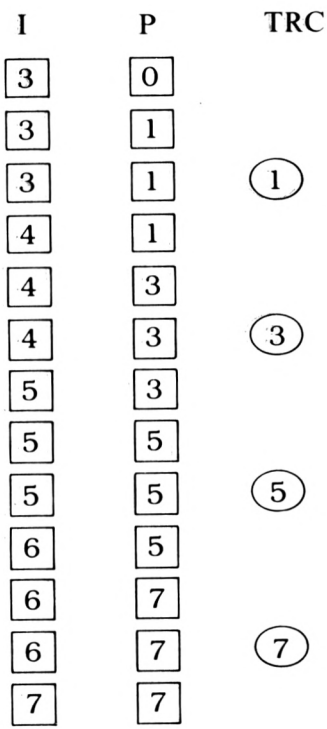
Ejercicios 1. Realice un diagrama de cajas para el siguiente programa:

```
100 DATA "CHINA","2","EUROPA"
110 FOR Q=1 TO 3
120 READ R$
130 PRINT R$
140 NEXT Q
150 END
```

2. Realice un diagrama de cajas para el programa siguiente:

```
20 FOR I=1 TO 10
30 LET K=2*I
40 IF K>4 THEN GOTO 60
50 NEXT I
60 PRINT "ESO ES TODO!"
70 END
```

3. Escriba un programa BASIC con una instrucción FOR-NEXT que se corresponda con el siguiente diagrama de cajas:



FORMA 2

La forma (2) es conocida como la instrucción FOR-NEXT-STEP. Trabaja de una manera similar, pero tiene la particularidad de variar el índice en valores distintos a 1.

La forma (2) de FOR-NEXT-STEP opera bajo las siguientes normas:

a) *Cuando STEP Q es un número positivo:*

1. La instrucción FOR K=M TO STEP Q da a K el valor M, y se ejecutan instrucciones BASIC hasta la instrucción NEXT K.
2. La instrucción NEXT K suma el valor Q a K, se detiene para ver si K es *mayor* que N, y si no lo es retorna el control a la instrucción siguiente a la sentencia FOR.
3. Una vez que la instrucción NEXT K incrementa K a un valor *mayor* que N, el programa seguirá en la primera instrucción después de la sentencia NEXT K.

b) *Cuando STEP Q es un número negativo:*

1. La instrucción FOR K=M TO N STEP Q da a K inicialmente el valor M, y se ejecutan las instrucciones BASIC hasta la sentencia NEXT K.
2. La instrucción NEXT suma el valor Q a K, se detiene a ver si K es *menor* que N y si no, retorna el control a la instrucción inmediatamente siguiente a la sentencia FOR.
3. Una vez que la sentencia NEXT K decrece K hasta un valor menor que N, el programa seguirá en la primera instrucción de después de la sentencia NEXT K.

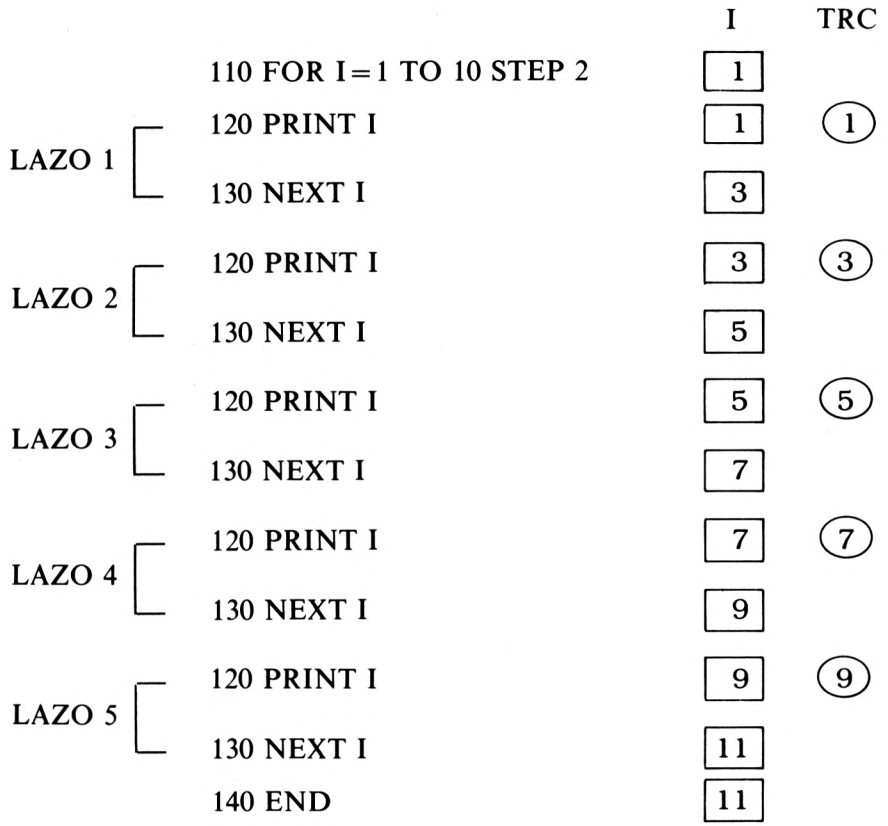
Los siguientes ejemplos pueden ayudar a clarificar ideas sobre la instrucción FOR-NEXT-STEP.

Ejemplo Este ejemplo muestra un incremento mayor que uno.

```
100 REM IMPRIME LOS NUMEROS IMPARES DE 1 A 9
105 REM Q ES POSITIVO 2
110 FOR I=1 TO 10 STEP 2
120 PRINT I
130 NEXT I
140 END
```

```
]RUN
 1
 2
 5
 7
 9
```

He aquí el diagrama de cajas:



Ejemplo El siguiente diagrama de cajas muestra un STEP negativo:

```
10 REM IMPRIME LOS NUMEROS 10,8,6
15 REM Q ES NEGATIVO 2
20 FOR J=10 TO 5 STEP -2
30 PRINT J
40 NEXT J
50 END
```

```
]RUN
10
8
6
```

He aquí el diagrama de cajas:

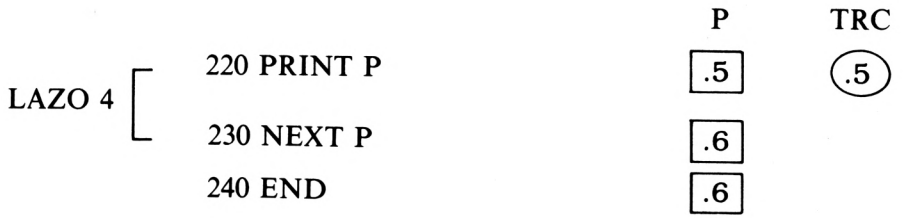
		J	TRC
	20 FOR J=10 TO 5 STEP -2	10	
LAZO	30 PRINT J	10	10
	40 NEXT J	8	
LAZO	30 PRINT J	8	8
	40 NEXT J	6	
LAZO	30 PRINT J	6	6
	40 NEXT J	4	
	50 END	4	

Ejemplo Este programa muestra un STEP que es una fracción decimal:

```
200 REM IMPRIME LOS NUMEROS .2,.3,.4,.5
210 FOR P=.2 TO .5 STEP .1
220 PRINT P
230 NEXT P
240 END
```

```
]RUN
.2
.3
.4
.5
```

		P	TRC
	210 FOR P=.2 TO .5 STEP .1	.2	
LAZO 1	220 PRINT P	.2	.2
	230 NEXT P	.3	
LAZO 2	220 PRINT P	.3	.3
	230 NEXT P	.4	
LAZO 3	220 PRINT P	.4	.4
	230 NEXT P	.5	



7.1

Ejercicios (cont.)

4. ¿Cuál es la solución de cada uno de estos programas?

a)
```
10 FOR K=1 TO 8
20 LET T=.2*K+1
30 PRINT T
35 NEXT K
40 END
```

b)
```
40 LET R=3.5
50 FOR U=R TO 0 STEP -.5
60 PRINT U,U*U
65 NEXT U
70 END
```

c)
```
100 LET E=3
110 FOR Q=E TO E+5 STEP E
120 PRINT Q
130 NEXT Q
140 END
```

d)
```
100 FOR J=8 TO 2
110 PRINT J
120 NEXT J
130 END
```
(¡cuidado con éste!)

Los siguientes programas muestran cómo puede hacerse la suma reiterativa utilizando un lazo FOR-NEXT.

Ejemplo
```
100 REM SUMA TODOS LOS NUMEROS PARES DE 2 A 100
110 LET S=0
120 FOR K=2 TO 100 STEP 2
```

```
130 LET S=S+K
140 NEXT K
150 PRINT "LA SUMA ES ";S
160 END
```

```
]RUN
LA SUMA ES 2550
```

Ejemplo

```
 30 REM SUMA TODOS LOS NUMEROS DE UNA SENTENCIA
    DATA
 40 DATA 34,56,35,45,-8
 50 DATA 2,34,445,67
 60 LET S=0
 70 FOR I=1 TO 9
 80 READ X
 90 LET S=S+X
100 NEXT I
110 PRINT "LA SUMA ES ";S
120 END
```

```
]RUN
LA SUMA ES 710
```

Ejemplo

```
100 REM SUMA DE UN NUMERO ARBITRARIO DE NUMEROS
110 INPUT "CUANTOS NUMEROS? ";Q
120 LET S=0
130 FOR I=1 TO Q
140 INPUT "INTRODUZCA UN NUMERO: ";N
150 LET S=S+N
160 NEXT I
170 PRINT S
180 END
```

```
]RUN
CUANTOS NUMEROS? 5
INTRODUZCA UN NUMERO: 10
INTRODUZCA UN NUMERO: 2
INTRODUZCA UN NUMERO: 1
INTRODUZCA UN NUMERO: 66
INTRODUZCA UN NUMERO: 4
83
```

Aplicaciones

Los siguientes programas son aplicaciones de la instrucción FOR-NEXT.

APLICACION

Este programa imprimirá en la pantalla el nombre del mes si se introduce el número de dicho mes:

```
100 REM HALLA EL MES
110 INPUT "INTRODUZCA EL NUMERO DEL MES DESEADO: ";N
120 IF N>12 OR N<1 THEN PRINT "LO SIENTO, ELECCION
    NULA.": GOTO 110
130 FOR I=1 TO N
140 READ M$
150 NEXT I
160 PRINT "ESE MES ES ";M$
170 DATA "ENERO","FEBRERO","MARZO","ABRIL"
180 DATA "MAYO","JUNIO","JULIO","AGOSTO","SEPTIEMBRE"
190 DATA "OCTUBRE","NOVIEMBRE","DICIEMBRE"
200 END
```

```
]RUN
INTRODUZCA EL NUMERO DEL MES DESEADO: 12
ESE MES ES DICIEMBRE
```

APLICACION

Suponga que el señor Solís invierte 12000 pesetas en una cuenta de ahorro por un período de diez años. Además suponga que el interés compuesto es del 9 por 100 anual.

El siguiente programa calcula e imprime el total de su cuenta al final de cada año.

```
10 REM INTERES COMPUESTO
20 LET P=12000
30 FOR I=1 TO 10
40 LET P=P+.09*P
45 PRINT "TOTAL DESPUES DE ";I;" AÑOS ES ";P
50 NEXT I
60 END
```

```
]RUN
TOTAL DESPUES DE 1 AÑOS ES 13080
TOTAL DESPUES DE 2 AÑOS ES 14257.2
TOTAL DESPUES DE 3 AÑOS ES 15540.348
TOTAL DESPUES DE 4 AÑOS ES 16938.9793
TOTAL DESPUES DE 5 AÑOS ES 18463.4875
TOTAL DESPUES DE 6 AÑOS ES 20125.2013
TOTAL DESPUES DE 7 AÑOS ES 21936.4695
TOTAL DESPUES DE 8 AÑOS ES 23910.7517
TOTAL DESPUES DE 9 AÑOS ES 26062.7194
TOTAL DESPUES DE 10 AÑOS ES 28408.3641
```

7.2

Ejercicios

1. Escriba un programa BASIC que calcule el capital total de una cuenta de ahorro después de un período de N años pagando un tipo de interés anual del R por 100 suponiendo una inversión inicial de P dólares.
2. Modifique el programa del problema 1 para calcular e imprimir también el importe total de los intereses devengados.

APLICACION

Es sabido que la suma de la serie infinita

$$\frac{1}{2} + \frac{1}{24} + \cdots + \frac{1}{K(K+1)(K+2)} + \cdots$$

es igual a 1. Aunque la suma infinita no es posible en el ordenador, podemos sumar tantos términos como deseemos, lo cual se hace en el siguiente programa:

```
25 REM SUMA DE MUCHOS TERMINOS
30 INPUT "CUANTOS TERMINOS QUIERE USTED? ";N
35 LET S=0
40 FOR K=1 TO N
50 LET S=S+1/(K*(K+1))
60 NEXT K
70 PRINT S
80 END
```

```
]RUN
CUANTOS TERMINOS QUIERE USTED? 500
.998003991
```

Ejercicios (cont.)

3. Escriba un programa que evalúe la suma infinita:

$$\frac{1}{2} + \frac{1}{24} + \cdots + \frac{1}{K(K+1)(K+2)} + \cdots$$

¿Cuál es la suma de: *a*) 10 términos; *b*) 100 términos; *c*) 1000 términos? ¿A qué valor cree usted que se aproximará la suma?

APLICACION

En el capítulo 1 introducimos los números factoriales. Se definen de la siguiente manera:

1!=1
2!=2
3!=3*2=6
4!=4*3*2=24
5!=5*4*3*2=120
6!=6*5*4*3*2=720
y así sucesivamente...

El siguiente programa calcula los factoriales:

```
100 REM ESTE PROGRAMA CALCULA N!
110 INPUT "INTRODUZCA N: ";N
120 LET S=1
130 FOR I=N TO 2 STEP -1
140 LET S=S*I
150 NEXT I
160 PRINT S
170 END
```

```
]RUN
INTRODUZCA N: 10
3628800
```

APLICACION

El siguiente programa calcula la media de los números 45, 33, 67, 88, 22, 45 e imprime los números que son mayores o iguales que la media:

```
100 DATA 45,33,67,88,22,45
105 LET S=0
```

```
110 FOR I=1 TO 6
120 READ X
130 LET S=S+X
140 NEXT I
150 LET A=S/6
160 RESTORE
170 FOR I=1 TO 6
180 READ X
190 IF (X>=A) THEN PRINT X
200 NEXT I
210 END
```

```
]RUN
67
68
```

7.2

Ejercicios (cont.)

4. Escriba un programa BASIC que calcule la media de los números 567, 45, 668, 332, −234, 44, 6789, −34, 67, 77 e imprima los números que sean mayores o iguales a la media.
5. Escriba un programa que genere e imprima 50 números enteros aleatorios entre 1 y 100 y su media.

APLICACION

La señorita Rodríguez guarda un listín telefónico en su ordenador personal. La lista es:

MARI JUANA	235-6666
LUISA GOMEZ	245-4545
ALICIA SAINZ	567-8989
MARIA CLARA	234-5678
FELIPE PUERTA	556-6767

El siguiente programa le permitirá dirigirse a la lista por el nombre:

```
100 REM LISTA DE TELEFONOS
110 DATA "MARI JUANA", "235-6666"
120 DATA "LUISA GOMEZ", "245-4545"
125 DATA "ALICIA SAINZ", "567-8989"
130 DATA "MARIA CLARA", "234-5678"
140 DATA "FELIPE PUERTA", "556-6767"
150 INPUT "CUAL ES EL NOMBRE? ";N$
```

```
160 FOR I=1 TO 5
170 READ X$,Y$
180 IF X$<>N$ THEN GOTO 200
190 PRINT "EL NUMERO DE TELEFONO ES ";Y$ :GOTO 220
200 NEXT I
210 PRINT "LO SIENTO, NOMBRE INEXISTENTE"
220 END
```

```
]RUN
CUAL ES EL NOMBRE? MARIA CLARA
EL NUMERO DE TELEFONO ES 234-5678
```

7.2

Ejercicios (cont.)

6. Modifique el programa «lista de teléfonos» de tal manera que pueda proporcionar un nombre dando un número de teléfono.

7. El programa es demasiado específico. Si se añaden más nombres y teléfonos a las sentencias DATA, no serán hallados. Reescriba el programa para que funcione aunque se le añadan más secuencias DATA.

7.3

Las instrucciones FOR-NEXT anidadas

Puesto que dentro de un lazo FOR-NEXT puede incluirse cualquier tipo de instrucciones BASIC, entonces dentro de una sentencia FOR-NEXT puede "anidarse" otra.

Ejemplo

```
100 FOR I=1 TO 6
110 FOR J=1 TO 5
115 PRINT I*J;" ";
120 NEXT J
130 PRINT
140 NEXT I
150 END
```

La salida en pantalla será:

```
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
4 8 12 16 20
5 10 15 20 25
6 12 18 24 30
```

7.3

Ejercicios

1. ¿Qué hace la línea 130?

2. Escriba un programa que imprima la siguiente tabla. Use sentencias FOR-NEXT anidadas.

1+1=2	1+2=3	1+3=4	1+4=5	1+5=6
2+1=3	2+2=4	2+3=5	2+4=6	2+5=7
3+1=4	3+2=5	3+3=6	3+4=7	3+5=8
4+1=5	4+2=6	4+3=7	4+4=8	4+5=9

7.4

Proyectos

1. *La serie de números Fibonacci:*

Los números Fibonacci son los siguientes:

1, 1, 2, 3, 5, 8, 13,...

Para generar esta lista de números, siga estos pasos:

a) Comenzando con los números 1 y 1...
b) Sume 1 + 1 y obtenga 2.
c) Sume 1 + 2 y obtenga 3.
d) Sume 2 + 3 y obtenga 5.
e) La norma general es sumar los últimos dos números de la serie para obtener el número siguiente. Así, en la secuencia anterior, el siguiente número después de 13 es 21.

Escribir un programa BASIC que genere e imprima los primeros N números de la serie Fibonacci, siendo N elegido por el usuario.

2. *Aproximación a π:*

Del capítulo 1 puede recordar que π, el número irracional 3.1415926..., puede ser aproximado por la suma de la serie infinita:

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \cdots$$

Escriba un programa BASIC para aproximar π sumando tantos términos como se desee.

3. *Financiación de un automóvil:*

Un automóvil es financiado en dólares a un interés mensual del R por 100. Suponiendo que el pago mensual es P dólares, escríbase un programa BASIC que:

a) Calcule e imprima el interés total pagado por K meses, y que
b) Calcule e imprima el saldo adeudado después de K meses.

4. *Problema del centro numérico:*

Un centro numérico es un número que separa una lista de números enteros (comenzando desde 1) en dos grupos de números cuyas sumas son iguales. El primer centro numérico es el 6, el cual separa la lista : 1, 2, 3, 4, 5, 6, 7, 8 en dos grupos (1, 2, 3, 4, 5) y (7, 8) cuyas sumas son ambas iguales a 15. El segundo centro numérico es 35. Separa una lista desde 1 a 49 en (1 a 34) y (de 36 a 49). La suma de los números de cada grupo es 595.

Escriba un programa que halle los centros numéricos entre 1 y N, siendo N elegido por el usuario.

5. *El número áureo:*

El número áureo es un número que tiene fascinados a muchos matemáticos, desde que los antiguos griegos obtuvieran su valor:

$(1 + \sqrt{5})/2 = 1.618033989...$

El número áureo está relativamente cerca de los números Fibonacci (ver proyecto 1). La razón de cualquier par de números Fibonacci consecutivos es aproximadamente igual al número áureo. La aproximación es mayor cuanto mayores sean los números Fibonacci usados. Escriba un programa BASIC que compruebe esta propiedad.

6. *Teoría de las probabilidades:*

El origen de la Teoría de las Probabilidades viene de este acontecimiento real:

> Chevalier de Méré escribió un carta a Pascal (1623-1662) donde le solicitaba la solución a un problema de juego. Méré jugó una partida de dados con un único dado, apostando que en cuatro tiradas del dado podía aparecer un 6 al menos una vez. Ganó este juego más a menudo que lo perdió. Más tarde cambió las normas del juego y tiró dos dados en vez de uno, apostando que al menos saldría un 6 doble en 24 tiradas de los dos dados. Para su sorpresa, perdió más veces que ganó (pensaba que la ventaja iba a ser la misma).

Escriba un programa BASIC que simule ambos juegos de dados. Debe obtener resultados similares.

PRACTICA 7.1

La instrucción FOR-NEXT

Introduzca la orden: NEW

Las instrucciones FOR-NEXT se usan para ejecutar una secuencia de sentencias un número finito de veces. Suponga que desea escribir un programa que visualice en la pantalla la cadena EL BASIC ES DIVERTIDO doce veces. Para efectuar esto la sentencia PRINT "EL BASIC ES DIVERTIDO" necesita ser ejecutada doce veces diferentes.

Introduzca el programa:

```
10 FOR K=1 TO 12 STEP 1
20 PRINT "EL BASIC ES DIVERTIDO"
30 NEXT K
40 END
```

Introduzca la orden: RUN

¿Cuántas veces está impresa la cadena EL BASIC ES DIVERTIDO? _____

La sentencia 10 FOR K=1 TO 12 STEP 1 hace tres cosas. Primero, inicializa el índice K a 1 (FOR K=1). Segundo, fija el valor límite de 12 para el índice K, y tercero, define un incremento de 1 que será usado más tarde por la sentencia NEXT K.

La instrucción NEXT K hace varias cosas. Primero, incrementa el índice según el valor de STEP. Segundo, se detiene a ver si el valor de K es mayor que 12; si no lo es, la instrucción NEXT K transfiere el control hacia atrás a la línea número 20. Finalmente, cuando K excede a 12, el control es pasado a la línea numerada a continuación de la sentencia NEXT K.

Las instrucciones FOR y NEXT junto con las sentencias entre ellas son comúnmente llamadas "lazo" o "bucle".

Introduzca el programa:

```
10 LET K=97
20 FOR K=2 TO 5 STEP 3
30 PRINT K
40 NEXT K
50 PRINT K
60 END
```

Introduzca la orden: RUN

¿Cuál es la respuesta del programa? _____

_____

_____

Fíjese que el primer resultado es 2 y no 97. Esto es porque la sentencia FOR inicializa el índice K a 2. El segundo resultado es 5 porque el índice K es incrementado en 3. El último resultado, 8, se imprimió en la línea número 50, no en la línea 30. Cuando el índice K fue incrementado de 5 a 8, se hizo mayor que el valor límite 5. La ejecución del programa se transfirió entonces a la línea 50.

Introduzca la orden: NEW

Introduzca el programa:
```
10 FOR K=1 TO 5
20 PRINT K
30 NEXT K
40 END
```

Introduzca la orden: RUN

¿Cuál es el resultado del programa? _____

_____

_____

¿Qué valor de incremento está usando el ordenador? _____

Cuando se excluye la instrucción STEP de la sentencia FOR, el incremento queda fijado en 1.

Introduzca la sentencia: 10 FOR K=15 TO 7 STEP −2

Introduzca la orden: LIST

Introduzca la orden: RUN

¿Cuál es la respuesta del programa? _____

_____

_____

¿Cuál es el valor del incremento para el índice? _____

_____

Usted debe haber escrito −2. El índice K es primero fijado en 15 y es incrementado en −2 hasta que K pasa el valor límite 7.

¿Cuál es el valor de K después de que se ha completado el lazo? _____

Usted debe haber escrito 5. Recuerde, el bucle no está completo hasta después de que el índice sobrepase el valor límite fijado por la sentencia FOR.

PRACTICA 7.2

Lazos anidados

Introduzca la orden: NEW

Siempre que un lazo FOR/NEXT esté encerrado dentro de otro lazo FOR/NEXT, esta disposición se llama anidamiento de lazos. Cuando sean usados los lazos aninados en programas, se deben observar dos normas:

1. El lazo interior debe estar totalmente encerrado dentro del lazo exterior.
2. Deben usarse variables índice diferentes para cada uno de los bucles FOR/NEXT.

Introduzca el programa:

```
10 FOR I=1 TO 3
20 PRINT "I= ";I
30 FOR K=1 TO 2
40 PRINT " K ES ";K
50 NEXT K
60 PRINT "**********"
70 NEXT I
80 END
```

Introduzca la orden: RUN

Fíjese que el bucle interior (líneas numeradas 30, 40 y 50) es ejecutado tres veces. Cuando I=1, I=2, y I=3.

¿Qué líneas numeradas pertenecen al lazo exterior? _____

Usted debe haber escrito las líneas números 10, 20, 30, 40, 50, 60 y 70. Los bucles conjuntados no pueden solaparse o cruzarse con otros.

Introduzca las sentencias: 50 NEXT I
70 NEXT K

Introduzca la orden: LIST

¿Qué líneas pertenecen al lazo que usa el índice I? _____

Usted debe haber escrito las líneas número 10, 20, 30, 40, 50.

¿Qué líneas pertenecen al lazo que usa el índice K? _____

Ahora debe haber escrito las líneas número 30, 40, 50, 60, 70.

Introduzca la orden: RUN

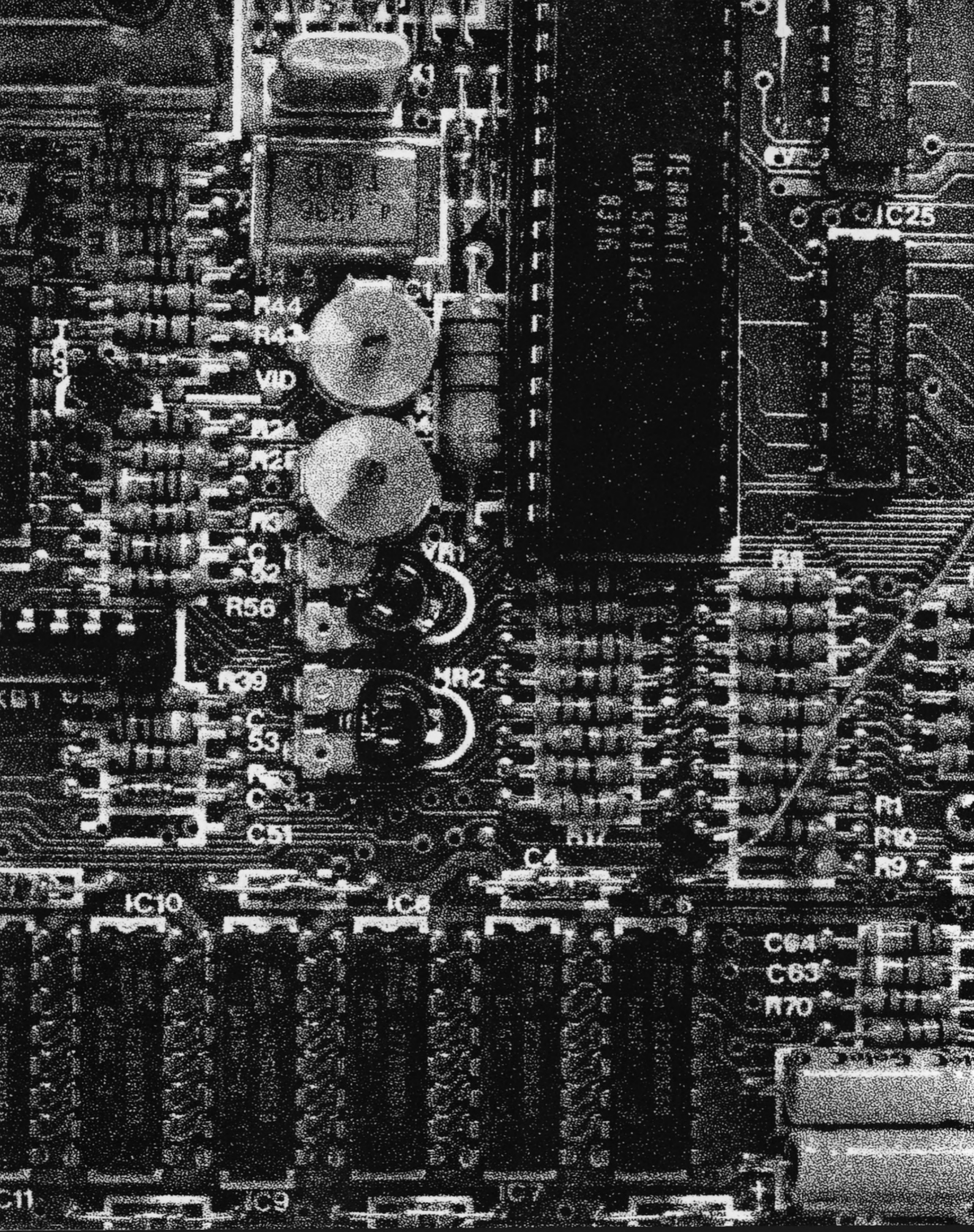
¿Cuál es la respuesta del ordenador? _____

Explique por qué se produce el error. _____

_____

El lazo que usa el índice K se solapa con el lazo que usa el índice I. Recuerde que el lazo interior debe estar totalmente encerrado dentro del lazo exterior. El error se originó porque los dos lazos estaban solapados.

_____


IC25
R44
VID
R56
R39
C51
C4
R1
R9
IC10
IC8
C64
C63
R70
IC9
IC7

8
Matrices

8.0
Introducción

Una reflexión sobre la forma de asignar nombres BASIC a las variables numéricas y de cadena podría revelarnos un aparente fallo. Puesto que el Applesoft sólo reconoce los dos primeros caracteres del nombre de una variable, y el primer carácter debe ser una letra, sólo hay 962 (=26*37) únicos nombres de variables (de hecho, son menos debido a las palabras reservadas). Esto parece sugerir que, durante el curso del funcionamiento de un programa sólo pueden ser almacenados al mismo tiempo en la memoria del ordenador menos de 962 valores diferentes de datos numéricos y menos de 962 valores diferentes de datos-cadena. Esto puede ser un tremendo obstáculo si consideramos que ciertas aplicaciones pueden necesitar el manejo de grandes cantidades de datos. Tal puede ser el caso de un programa que calcule la nómina de una gran compañía o un programa que calcule y almacene los 1000 primeros números primos. Así pues, ¿cómo salvaremos este obstáculo? La respuesta está en el concepto de matriz. Las matrices son también de dos tipos: *numéricas* y de *cadena*, y tienen variedad de tamaños. Tomaremos primero las matrices numéricas.

8.1

Matrices numéricas unidimensionales

Una matriz numérica es una colección de variables numéricas que tienen todas el mismo nombre, pero que se distinguen entre ellas por el uso de *índices*. Un índice es un número entre paréntesis que sigue al nombre de la matriz.

Ejemplo A(0), A(1), A(2), A(3), . . . , A(998), A(999), A(1000)
A6(0), A6(1), A6(2), A6(3), . . . , A6(8), A6(9), A6(10)

En este ejemplo la matriz A puede almacenar 1001 diferentes datos numéricos. Digamos también que hay 1001 celdas en una matriz. La matriz A6 puede almacenar once secciones diferentes de datos numéricos.

NORMAS PARA EL USO DE LAS MATRICES NUMERICAS

1. El nombre de la matriz es el de cualquier variable numérica válida.
2. Cada índice debe ser un número entero (por ejemplo, 0, 1, 2, ...) o cualquier variable o expresión numéricas que evalúen un número entero. Los índices que no sean enteros se redondean a enteros.

Se pueden asignar valores numéricos a estas matrices utilizando las instrucciones LET, READ e INPUT. El siguiente es un ejemplo simple que usa la instrucción LET:

Ejemplo

	A(3)	B(5)	J
	0	0	0
LET A(3)=4	4	0	0
LET J=3	4	0	3
LET A(J)=5	5	0	3
LET B(J+2)=6	5	6	3
LET B(A(J))=8	5	8	3

Ejemplo El siguiente programa BASIC almacenará y sumará los números 4, 9 y 8:

	C(1)	C(2)	C(3)	S	TRC
	0	0	0	0	
100 LET C(1)=4	4	0	0	0	
110 LET C(2)=9	4	9	0	0	
120 LET C(3)=8	4	9	8	0	
130 LET S=C(1)+C(2)+C(3)	4	9	8	21	
140 PRINT S	4	9	8	21	21
150 END					

```
]RUN
21
```

Puede escribirse el mismo programa utilizando READ/DATA:

```
100 DATA 4,9,8
110 READ C(1),C(2),C(3)
120 LET S=C(1)+C(2)+C(3)
130 PRINT S
140 END
```

El mismo programa puede escribirse usando un lazo FOR-NEXT de esta manera:

```
 90 REM ASIGNAR DATOS A UNA MATRIZ
100 DATA 4,9,8
110 FOR I=1 TO 3
120 READ C(I)
130 NEXT I
135 REM CALCULA LA SUMA
140 LET S=0
145 FOR I=1 TO 3
150 LET S=S+C(I)
160 NEXT I
165 REM IMPRIME LA SUMA
170 PRINT S
180 END
```

```
]RUN
21
```

Ejemplo El programa anterior puede generalizarse mediante asignación de los datos con una instrucción INPUT:

```
100 REM ASIGNA DATOS A LA MATRIZ
110 FOR I=1 TO 3
120 INPUT "INTRODUZCA UN NUMERO: ";C(I)
130 NEXT I
135 REM CALCULA LA SUMA
140 LET S=0
150 FOR I=1 TO 3
160 LET S=S+C(I)
170 NEXT I
180 REM IMPRIME LA SUMA
190 PRINT "LA SUMA ES ";S
200 END
```

```
]RUN
ESCRIBA UN NUMERO: 23
ESCRIBA UN NUMERO: 5
ESCRIBA UN NUMERO: -22
LA SUMA ES 6
```

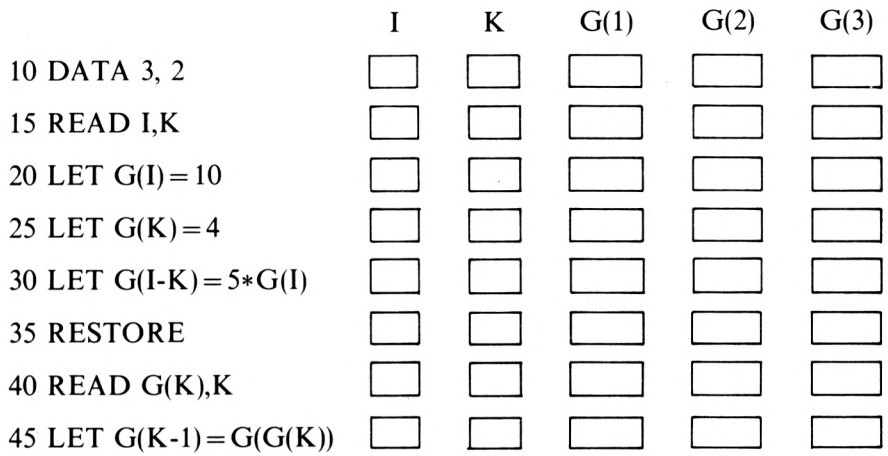
8.1 Ejercicios

1. Realice un diagrama de cajas para el programa del último ejemplo.

2. Escriba las instrucciones LET correctas para el siguiente diagrama de cajas con la restricción de que *solamente* pueden usarse a la derecha de las sentencias LET las variables de matriz, después de las primeras dos sentencias:

	A(1)	A(2)	A(3)	A(4)
	0	0	0	0
LET A(2)=4	0	4	0	0
LET A(4)=6	0	4	0	6
	.5	4	0	6
	.5	1	0	6
	.5	1	0	8
	.5	1	9	8
	.5	0	9	8

3. Realice un diagrama de cajas para el siguiente programa:



4. Escriba un programa que halle la media de diez números e imprima los números en dos grupos: los menores o iguales a la media, y los mayores que la media. Los diez números deben poder ser elegidos por un usuario (INPUT) y almacenados en una matriz.

8.2

La instrucción DIMensión

Con el fin de almacenar números en una matriz el ordenador debe ser informado de cuántos elementos tendrá la matriz. Esto se hace con una instrucción DIMensión. Si la matriz se llama X, la forma de la instrucción dimensión es

```
DIM X(N)
```

donde N es un número entero, una variable, o una fórmula que evalúa un número entero.

Si se necesita dimensionar varias matrices, pueden incluirse en la misma sentencia DIM, tal como:

```
DIM G(100), Y(2*N+1), Z(K)
```

Si no se van a usar más que once celdas numeradas de 0 a 10, entonces la matriz no tiene que ser dimensionada.

Ejemplo He aquí un programa que almacena tres notas de examen por cada uno de los treinta estudiantes de una clase:

```
100 DIM E1(30), E2(30), E3(30)
110 REM ALMACENA NOTAS
120 FOR I=1 TO 30
130 PRINT "ESTUDIANTE NUMERO ";I
140 INPUT "CUALES SON LAS NOTAS? ";E1(I),E2(I),E3(I)
145 PRINT
150 NEXT I
160 END
```

```
]RUN

ESTUDIANTE NUMERO 1
CUALES SON SUS NOTAS? 56,78,49

ESTUDIANTE NUMERO 2
CUALES SON SUS NOTAS? 78,85,93

ESTUDIANTE NUMERO 3
CUALES SON SUS NOTAS? 95,98,88

(etc.)
```

8.2 Ejercicios

1. Haga un programa que

   a) Calcule la nota media de exámenes por cada estudiante y almacene las medias en A(1), A(2), ..., A(30).

   b) Imprima en la pantalla el número de los estudiantes (es decir, 1, 2, 3, ..., 30) cuya media sea superior a 80.

Ejemplo Los siguientes pasos calculan la desviación típica de un conjunto dado de valores de matriz:

a) Calcule la media de los valores de matriz.
b) Reste esta media de cada uno de los valores de la matriz.
c) Eleve al cuadrado cada uno de los valores de *b*).
d) Calcule la media de los valores de *c*).
e) Haga una raíz cuadrada de la media de *d*).

El siguiente programa permitirá hallar la desviación típica de veinticinco valores de matriz:

```
100 DIM A(25),B(25)
110 REM INTRODUCE 25 NUMEROS EN LA MATRIZ A
120 FOR I=1 TO 25
130 INPUT A(I)
140 NEXT I
150 REM REALIZA EL PASO (A)
155 LET S=0
160 FOR I=1 TO 25
170 LET S=0
180 NEXT I
190 REM USAMOS A PARA LA MEDIA. LA VARIABLE SIMPLE A ES
200 REM DIFERENTE DE LA MATRIZ A
210 LET A=S/25
220 REM REALIZA EL PASO (B)
230 FOR J=1 TO 25
240 LET B(J)=A(J)-A
250 NEXT J
260 REM REALIZA EL PASO (C)
270 FOR J=1 TO 25
280 LET B(J)=B(J)/2
290 NEXT J
300 REM REALIZA EL PASO (D)
310 LET S=0
320 FOR I=1 TO 25
330 LET S=S+B(I)
340 NEXT I
350 LET S=S/25
360 REM COMPLETA EL PASO (E)
370 LET S=SQR(S)
380 REM EXHIBE LA DESVIACION TIPICA
390 PRINT S
400 END
```

Ejemplo El siguiente programa BASIC almacenará 200 números aleatorios en la matriz W, calcula su media, imprime todos los números que sean mayores que la media, y dice cuántos hay:

```
100 REM ALMACENAMIENTO Y PROMEDIO DE 200 NUMEROS
    ALEATORIOS
110 DIM W(200)
```

```
120 LET S=0
130 LET C=0
140 FOR K=1 TO 200
150 LET W(K)=RND(1)
160 LET S=S+W(K)
170 NEXT K
180 LET A=S/200
190 PRINT "LA MEDIA ES ";A
200 FOR K=1 TO 200
210 IF (W(K)>A) THEN PRINT W(K) :LET C=C+1
220 NEXT K
230 PRINT C;" NUMEROS QUE SON MAYORES QUE LA MEDIA."
240 END
```

8.2

Ejercicios (cont.)

2. ¿Por qué razón está (W(K)>A) en vez de W(K)>A en la línea 210?

Si el programador desea reservar un número arbitrario de celdas en una matriz, entonces la sentencia DIM puede ser escrita de la forma:

```
DIM A(N)
```

donde N es cualquier variable numérica o expresión aritmética. *Precaución:* Una vez que una matriz está dimensionada, no puede ser redimensionada, a menos que sea dada la orden CLEAR.

Ejemplo El siguiente programa almacenará cualquier cantidad de números aleatorios:

```
100 INPUT "¿CUANTOS NUMEROS ALEATORIOS
    ALMACENARA? ";N
110 DIM A(N)
120 FOR K=1 TO N
130 LET A(K)=RND(1)
140 NEXT K
150 END
```

8.2

Ejercicios (cont.)

3. Escriba un programa que:

   a) Almacene N números aleatorios entre 1 y 100.

   b) Calcule e imprima su media.

c) Imprima todos los números mayores que la media.

d) Imprima todos los números pares de la lista.

Nota para *d*): cuando un número par es dividido por 2 el resto es 0. Use la función INT para ayudarle a comprobar esta condición.

4. El programa de «almacenamiento y promedio de 200 números aleatorios» necesita ser generalizado. Reescríbalo de tal manera que pueda ser almacenada y promediada cualquier cantidad de números aleatorios.

5. Modifique el programa de desviación típica de tal manera que pueda ser usado un arbitrario número de valores.

8.3

Ordenamiento

El ordenamiento es la disposición de los datos en orden ascendente o descendente. Este juega un importante papel en las aplicaciones del ordenador, particularmente en los negocios.

Ejemplo

DATOS DESORDENADOS: 45, 23, 78, 100, 2, 45, 2
ORDENAMIENTO ASCENDENTE: 2, 2, 23, 45, 45, 78, 100
ORDENAMIENTO DESCENDENTE: 100, 78, 45, 45, 23, 2, 2

Hay muchas técnicas para el ordenamiento. En este libro discutiremos uno de los más populares métodos de ordenación (aunque también es uno de los más lentos).

Para colocar cuatro números en orden ascendente usando este método, se usará el siguiente procedimiento:

1. Almacene los cuatro números en la matriz A.
2. Compare los valores de las dos primeras celdas. Si A(1) es menor que A(2), intercambie los dos valores. Luego compare A(2) y A(3), e intercámbielos si A(2) es menor que A(3). Luego compare A(3) con A(4), e intercámbielos si es necesario. A esto se le llama *pase*.
3. Repita dos veces más el proceso 2.
4. Imprima la matriz en la pantalla.

Los procesos 2 y 3 pueden ser comprendidos mejor con la tabla siguiente. Suponga que los cuatro números que queremos ordenar son 2, 8, 5, −1. Nuestro objeto es ordenarlos de menor a mayor.

RELACION	¿V/F?	¿INTERCAMBIO?	A(1)	A(2)	A(3)	A(4)
			2	8	5	−1
PASO 1:						
A(1) > A(2)	FALSA	NO	2	8	5	−1
				intercambio		
A(2) > A(3)	VERDAD	SI	2	5	8	−1
					intercambio	
A(3) > A(4)	VERDAD	SI	2	5	−1	8
PASO 2:						
A(1) > A(2)	FALSA	NO	2	5	−1	8
				intercambio		
A(2) > A(3)	VERDAD	SI	2	−1	5	8
A(3) > A(4)	FALSA	NO	2	−1	5	8
PASO 3:						
			intercambio			
A(1) > A(2)	VERDAD	SI	−1	2	5	8
A(2) > A(3)	FALSA	NO	−1	2	5	8
A(3) > A(4)	FALSA	NO	−1	2	5	8

Partiendo de esta tabla, escribimos el siguiente programa BASIC:

```
100 REM ORDEN ASCENDENTE
110 DIM A(4)
120 DATA 2,8,5,—1
130 REM ALMACENA NUMEROS EN LA MATRIZ A
140 FOR I=1 TO 4
150 READ A(I)
160 NEXT I
170 REM **********************
180 REM PASO UNO
190 FOR J=1 TO 3
200 IF A(J)>A(J+1) THEN LET L=A(J) :LET A(J)=A(J+1) :
      LET A(J+1)=L
210 NEXT J
220 REM **********************
230 REM PASO DOS
240 FOR J=1 TO 3
250 IF A(J)>A(J+1) THEN LET L=A(J) :LET A(J)=A(J+1) :
      LET A(J+1)=L
260 NEXT J
270 REM **********************
280 REM PASO TRES
290 FOR J=1 TO 3
300 IF A(J)>A(J+1) THEN LET L=A(J) :LET A(J)=A(J+1) :
      LET A(J+1)=L
310 NEXT J
320 REM LOS PASOS ESTAN HECHOS, AHORA IMPRIME LA
      MATRIZ
330 FOR I=1 TO 4
340 PRINT A(I);" ";
350 NEXT I
360 END
```

```
]RUN
—1 2 5 8
```

El lector atento se dará cuenta que ¡las tres pasadas son idénticas! Esta repetición puede ser eliminada del programa insertando el lazo FOR-NEXT, el cual hará las pasadas dentro de otra sentencia FOR-NEXT que funcionará tres veces. A esto se le llama *anidamiento de lazos FOR-NEXT*.

El siguiente programa muestra el anidamiento de lazos:

```
100 REM ORDEN ASCENDENTE
115 DIM A(4)
120 DATA 2,8,5,—1
130 FOR I=1 TO 4
140 READ A(I)
150 NEXT I
155 REM EL LAZO "K" REPITE EL PASO TRES VECES
160 FOR K=1 TO 3
170 FOR J=1 TO 3
190 IF A(J)>A(J+1) THEN LET L=A(J) :LET A(J)=A(J+1) :
      LET A(J+1)=L
230 NEXT J
240 NEXT K
250 REM EL ORDENAMIENTO ESTA HECHO, AHORA IMPRIME
      LA MATRIZ
260 FOR I=1 TO 4
270 PRINT A(I);" ";
280 NEXT I
290 END
```

```
]RUN
—1 2 5 8
```

8.3

Ejercicios

1. Se requieren tres pasadas para ordenar cuatro números, ¿Cuántas pasadas son necesarias para ordenar seis números? ¿Cuántas pasadas son necesarias para ordenar N números?

2. Modifique el programa de ordenación de tal manera que ordene los números:

   23, −56, 100, 45, 56, 34, 444, 67, 0, 45

   en orden descendente.

3. Si fueran borradas del primer programa de ordenamiento, las líneas 220 a 310, ¿qué podría efectuar el programa?

4. Escriba un ordenamiento de tal manera que los números que vayan a ser ordenados sean arbitrarios en número e introducidos por una sentencia INPUT.

5. El anterior método de ordenamiento requiere (N – 1)*(N – 1) comparaciones de celdas adyacentes de una matriz que contenga N elementos (esto es así porque hay N – 1 pasadas con N – 1 comparaciones por pasada). Escriba un programa para acortar el proceso, de tal manera que sólo se requieran N*(N – 1)/2 comparaciones.

   NOTA: Después que se ha hecho la primera pasada el mayor valor está en A(N) (suponiendo que se ha ordenado en sentido ascendente). No hay realmente necesidad de otras pasadas para seguir comparando este mayor valor con los otros valores.

8.4

Aplicaciones

APLICACION: CONVERSION DE NUMEROS DECIMALES EN BINARIOS

El sistema de números binarios juega un importante papel en la operación interna de los ordenadores. Los números binarios están formados con los dígitos cero (0) y uno (1).

Ejemplo

11001 en binario es igual que 25 en decimal
111101 en binario es igual que 61 en decimal
1010101 en binario es igual que 85 en decimal

Los números enteros decimales pueden convertirse en números binarios dividiendo repetidamente por 2 y almacenando el resto. El siguiente ejemplo demuestra cómo se hace esto.

EJEMPLO DE CONVERSION DE NUMEROS DECIMALES EN NUMEROS BINARIOS

Convertir 25 en un número binario:

25 dividido entre 2 es 12 con un resto de 1
12 dividido entre 2 es 6 con un resto de 0
6 dividido entre 2 es 3 con un resto de 0
3 dividido entre 2 es 1 con un resto de 1
1 dividido entre 2 es 0 con un resto de 1

Los restos son leídos desde abajo hacia arriba para formar el binario equivalente de 25. Así, el número decimal 25 es equivalente al número binario 11001.

El siguiente programa convertirá los números enteros decimales en binarios:

```
100 REM CONVERSION DE N DE DECIMAL EN BINARIO
115 DIM R(100)
120 REM MODULO DE ASIGNACION
125 LET I=0
130 INPUT "INTRODUZCA UN NUMERO DECIMAL: ";N
140 LET I=I+1
145 REM MODULO DE CALCULO. R(I) ALMACENA EL RESTO
150 LET R(I)=N-2*INT(N/2)
160 LET N=INT(N/2)
170 IF N>0 THEN GOTO 140
180 REM MODULO DE SALIDA
190 FOR K=I TO 1 STEP -1
200 PRINT R(K);
210 NEXT K
220 END
```

```
]RUN

INTRODUZCA UN NUMERO DECIMAL: 100
1100100
```

8.4

Ejercicios

1. Realice un diagrama de cajas para el programa anterior cuando el valor de N es 6.

2. Modifique el programa de conversión de tal manera que convierta cualquier número entero a la base 8.

3. Modifique el programa de conversión de tal manera que convierta cualquier número decimal entero a la base 5 o base 7, dependiendo del deseo del usuario.

APLICACION: BUSQUEDA

Un reciente estudio ha mostrado que las personas que duermen una media de más de siete horas cada noche vive un número más corto de

años. La siguiente tabla contiene estadísticas del sueño y longevidad de 8 personas fallecidas:

Media del número de horas dormidas por la noche	Años vividos
6	67
6.5	72
5.75	80
9	55
8.5	62
5	81
6	70
9	50

En muchas aplicaciones de la programación es necesario extraer valores específicos de una matriz.

Ejemplo El siguiente programa BASIC hallará, de la tabla de longevidad, la media del número de años vividos por aquellos que durmieron más de 6.5 horas por noche:

```
100 DATA 6, 67, 6.5, 72, 5.75, 80, 9, 55
110 DATA 8.5, 62, 5, 81, 6, 70, 9, 50
120 DIM S(8),Y(8)
122 FOR K=1 TO 8
124 READ S(K), Y(K)
126 NEXT K
128 LET T=0
129 LET C=0
130 REM EL LAZO FOR-NEXT BUSCARA EN LA MATRIZ S
150 FOR I=1 TO 8
160 IF S(I)<=6.5 THEN GOTO 170
162 LET C=C+1
165 LET T=T+Y(I)
168 PRINT S(I),Y(I)
170 NEXT I
175 PRINT "MEDIA DE AÑOS VIVIDOS: ";T/C
180 END
```

```
]RUN

9                55
8.5              62
9                50
MEDIA DE AÑOS VIVIDOS: 55.6666667
```

Ejercicios (cont.)

4. Modifique el programa anterior de forma que busque sólo los valores de la matriz de aquellos individuos que durmieron más de 6.5 horas cada noche, pero que no vivieron más allá de los setenta años.

5. Escriba un programa que imprima sólo los valores de matriz de aquellos que vivieron más de sesenta y cinco años, y que además halle la media del tiempo dormido por las personas que entren dentro de esta categoría.

6. Escriba un programa BASIC que genere y almacene 50 números enteros aleatorios entre 1 y 20 y cuente e imprima el número de dieces y cincos que hay en la lista.

APLICACION: ELECCION ALEATORIA

Ejemplo de elección con reemplazamiento

La facultad de realizar una elección entre un conjunto fijo de datos es importante en muchas disciplinas. El siguiente programa elegirá aleatoriamente con reemplazamiento tres valores de la lista de datos 34, 67, 345, 35, 678. Elegir con reemplazamiento significa que cada número se restituye a la lista después de haber sido elegido y, por tanto, puede ser elegido más de una vez.

```
100 REM ELECCION DE 3 VALORES ALEATORIOS
115 DIM A(5)
120 DATA 34,67,345,35,678
130 FOR I=1 TO 5
140 READ A(I)
150 NEXT I
155 FOR I=1 TO 3
160 REM R ES UN INDICE ALEATORIO PARA LA MATRIZ
180 LET R=INT(5*RND(1))+1
190 PRINT A(R)
200 NEXT I
210 END
```

```
]RUN

345
67
345
```

Ejercicios (cont.)

7. Escriba un programa que haga treinta elecciones aleatorias diferentes, y que cada elección conste de tres números. Use los datos anteriores.

8. Escriba un programa que extraiga treinta números aleatorios del 1 al 365. Si sucede que cualquier número es extraído más de una vez, el programa imprimirá en la pantalla "EXITO". Si los treinta números extraídos son diferentes, el programa escribirá "FRACASO".

9. Si dos o más personas de un grupo de treinta tienen el mismo cumpleaños (día y mes) a esto se le llama "EXITO". ¿Ve usted que el problema 9 es una simulación de este acontecimiento? Un famoso resultado en la Teoría de las Probabilidades, conocido como "el problema del cumpleaños", muestra que la probabilidad de dos personas, de un grupo de treinta, de tener el mismo cumpleaños, es aproximadamente del 70 por 100. Compruebe esta afirmación escribiendo un programa que cuente el número de éxitos en 100 ensayos del programa del problema 9. Serán aproximadamente setenta.

Ejemplo de elección sin reemplazamiento

El siguiente programa elegirá los valores del último programa sin reemplazamiento. Esto significa que una vez que un número es seleccionado no puede volver a ser seleccionado de nuevo.

```
100 REM ELECCION DE 3 VALORES AL AZAR SIN RESTITUCION
120 DIM A(5)
130 DATA 34, 67, 345, 35, 678
140 FOR I=1 TO 5
150 READ A(I)
160 NEXT I
170 FOR I=1 TO 3
180 LET L=INT(5*RND(1))+1
185 IF A(L)=-999 THEN GOTO 180
190 PRINT A(L)
195 REM -999 ES UNA SEÑAL
200 LET A(L)=-999
205 NEXT I
210 END
```

```
]RUN
34
35
345
```

8.4

Ejercicios (cont.)

11. Escriba un programa que almacene los números enteros de 1 a 80 en una matriz, y que haga una elección aleatoria de veinte de los números, sin reemplazamiento, y los imprima. ¿Qué situación "del mundo real" refleja este modelo? Nota: piense apostando.

8.5

Matrices numéricas bidimensionales

Imagine un rectángulo de cajas, dividido en tres filas por cinco columnas. Las tres filas están numeradas 0, 1 y 2 y las cinco columnas están numeradas 0, 1, 2, 3 y 4. Cada caja puede ser únicamente localizada diciendo qué fila y qué columna caen en ella. Esto es exactamente la idea que hay detrás de una matriz bidimensional, también denominada "tabla". Las quince celdas de la matriz de tres por cinco son ahora variables individuales, siendo cada una capaz de almacenar datos numéricos. Suponga que la matriz se llama T. La sentencia

```
LET T(2,4)=9
```

asigna el número 9 a la caja de la fila 2 y la columna 4. Tenga en mente que la disposición es siempre: primero la fila, segundo la columna (las columnas van de arriba a abajo, como las columnas de un templo griego). Las sentencias:

```
LET T(2,4)=9
LET T(2,1)=8
LET T(1,2)=7
LET T(0,0)=6
```

pueden producir el siguiente diagrama de cajas:

Matriz T

Filas	Columnas 0	1	2	3	4
0	6				
1			7		
2		8			9

Obsérvese que T(2,1) no es la misma variable que T(1,2) y que T(3,4) no tiene sentido en nuestro ejemplo, ya que sólo tenemos tres filas numeradas: 0, 1, y 2.

Si ambos subíndices, que indican las filas o columnas, pasan de diez, entonces la columna debe ser DIMensionada. Una matriz U con cuatro filas y trece columnas podrían ser fijadas al principio del programa escribiendo

```
10 DIM U(3,12)
```

Recuerde que puesto que hay una fila cero y una columna cero, la sentencia DIM anterior era realmente una matriz con cuatro filas y trece columnas.

Ejemplo

El siguiente programa almacena los números dentro de una matriz bidimensional K. La fila cero y la columna cero no son usadas en este ejemplo, que es totalmente típico en las aplicaciones.

```
10 LET K(1,2)=2
20 LET K(2,2)=1
```

Matriz K

fila \ columna	1	2
1	0	2
2	0	0

fila \ columna	1	2
1	0	2
2	0	1

30 LET K(K(1,2),1)=4

fila \ columna	1	2
1	0	2
2	4	1

40 LET K(1,1)=K(1,2)*K(2,1)

fila \ columna	1	2
1	8	2
2	4	1

8.5

Ejercicios

1. Rellene la matriz P según el siguiente programa:

```
10 DATA 45,56,2,5
20 FOR I=1 TO 2
30 FOR J=1 TO 2
40 READ P(I,J)
50 NEXT J :NEXT I
60 END
```

Ejemplo Los recibos de la luz del señor Orzán de enero a diciembre fueron 3100, 2400, 2300, 2300, 2100, 2400, 2600, 3400, 3300, 2500, 2800 y 3000 pesetas, respectivamente. Sus recibos del gas de los mismos meses fueron 2800, 2500, 1800, 1500, 1300, 1400, 1200, 1000, 1000, 1700, 2200 y 2700 pesetas. Sus recibos de teléfono durante todo el año fueron 5200, 6100, 4800, 9900, 4100, 3800, 6500, 4000, 4500, 5700, 5300 y 6500 pesetas. Una matriz bidimensional almacenará todos los recibos del señor Orzán. Las columnas representan los meses, y las filas son los tipos de factura. El siguiente programa BASIC leerá y almacenará los recibos del señor Orzán.

```
10 DIM U(3,12)
20 FOR R=1 TO 3
30 FOR C=1 TO 12
```

```
 40 READ U(R,C)
 50 NEXT C
 60 NEXT R
 70 REM DATOS DE LAS FACTURAS ELECTRICAS
 80 DATA 3100, 2400, 2300, 2300, 2100, 2400, 2600, 3400, 3300, 2500,
    2800, 3000
 90 REM DATOS DE LAS FACTURAS DEL GAS
100 DATA 2800, 2500, 1800, 1500, 1300, 1400, 1200, 1000, 1000, 1700,
    2200, 2700
110 REM DATOS DE LAS FACTURAS DEL TELEFONO
120 DATA 5200, 6100, 4800, 9900, 4100, 3800, 6500, 4000, 4500, 5700,
    5300, 6500
```

Considere ahora la siguiente aplicación. El señor Orzán quiere poder conocer sus gastos domésticos totales de cualquier mes del año. Escriba el resto del programa que hemos empezado anteriormente:

```
130 INPUT "QUE MES (DEL 1 AL 12)? ";M
140 LET S=0
150 FOR I=1 TO 3
160 LET S=S+U(I,M)
170 NEXT I
180 PRINT "EL GASTO DE ESE MES FUE DE ";S
190 INPUT "LO INTENTA OTRA VEZ? ";A$
200 IF A$="SI" THEN GOTO 130
210 END
```

```
]RUN

QUE MES (DEL 1 AL 12)? 10
EL GASTO DE ESE MES FUE DE 9900
LO INTENTA OTRA VEZ? NO
```

8.5

Ejercicios (cont.)

2. Añada al programa del señor Orzán la facultad de calcular el total de los gastos anuales para cualquier tipo de factura doméstica seleccionada. (Nota: la fila está fija, pero los meses cambian del 1 al 12.)

3. Añada una característica al programa de tal manera que el señor Orzán pueda solicitar al ordenador que exponga en la pantalla todos los recibos del gas que sobrepasen una cantidad que él introduzca.

4. Modifique el programa del problema 3 de tal manera que el señor Orzán pueda también elegir los gastos domésticos que él quiera.

Ejemplo El señor Ruiz desea almacenar en el ordenador las M notas de examen de cada uno de los N estudiantes de su clase de estadística:

```
 90 REM N ES EL NUMERO DE ESTUDIANTES
 95 REM M ES EL NUMERO DE NOTAS DE EXAMEN POR
    ESTUDIANTE
100 INPUT "CUANTOS ESTUDIANTES TIENE USTED? ";N
110 INPUT "CUANTOS EXAMENES POR CADA
    ESTUDIANTE? ";M
120 DUM A(N,M)
130 FOR I=1 TO M
140 PRINT "NUMERO DE ESTUDIANTE ";I
150 FOR J=1 TO M
160 INPUT "ESCRIBA UNA NOTA DE EXAMEN: ";A(I,J)
170 NEXT J
180 NEXT I
190 END
```

```
]RUN

CUANTOS ESTUDIANTES TIENE USTED? 3
CUANTOS EXAMENES POR CADA ESTUDIANTE? 2
ESTUDIANTE NUMERO 1
ESCRIBA UNA NOTA DE EXAMEN: 70
ESCRIBA UNA NOTA DE EXAMEN: 75
ESTUDIANTE NUMERO 2
ESCRIBA UNA NOTA DE EXAMEN: 80
ESCRIBA UNA NOTA DE EXAMEN: 67
ESTUDIANTE NUMERO 3
ESCRIBA UNA NOTA DE EXAMEN: 89
ESCRIBA UNA NOTA DE EXAMEN: 98
```

8.5

Ejercicios (cont.)

5. El señor Ruiz necesita desarrollar este programa tal que pueda calcular la nota media de cada uno de sus estudiantes y la media de la clase para cada examen. Desarrolle el programa por él.

Ejemplo El señor Masia, un famoso vendedor de bienes inmuebles, utiliza un ordenador para registrar sus ventas. El siguiente programa hallará el total de sus comisiones y el valor total de las casas que vendió:

```
100 INPUT "INTRODUZCA EL NUMERO DE CASAS
    VENDIDAS: ";T
110 DIM N(T,2)
120 FOR I=1 TO T
```

```
130 PRINT "NUMERO DE LA CASA: ";I
140 INPUT "CUAL FUE LA COMISION? ";N(I,1)
150 INPUT "CUAL FUE EL PRECIO DE LA CASA? ";N(I,2)
160 NEXT I
170 REM SUMA DE LAS COMISIONES
175 LET S=0
180 FOR I=1 TO T
190 LET S=S+N(I,1)
200 NEXT I
210 PRINT "LA COMISION TOTAL ES ";S
220 LET S=0
230 REM HALLA EL VALOR TOTAL DE LAS CASAS
240 FOR I=1 TO T
250 LET S=S+N(I,2)
260 NEXT I
270 PRINT "EL VALOR TOTAL DE LAS VENTAS DE CASAS ES ";S
280 END
```

```
]RUN

INTRODUZCA EL NUMERO DE CASAS VENDIDAS: 2
NUMERO DE LA CASA: 1
CUAL FUE LA COMISION? 6000
CUAL FUE EL PRECIO DE LA CASA? 100000
NUMERO DE LA CASA: 2
CUAL FUE LA COMISION? 12000
CUAL FUE EL PRECIO DE LA CASA? 250000
LA COMISION TOTAL ES 18000
EL VOLUMEN TOTAL DE LAS VENTAS DE CASAS ES 350000
```

8.6

Matrices de cadena uni y bidimensionales

Las versiones populares del BASIC halladas en microordenadores siempre tienen la facultad de almacenar datos de cadena en las celdas de una matriz, conocida como *matriz tipo cadena.* Estas tienen nombres de variable de cadena típicos y siguen las mismas normas observadas para las matrices numéricas. Cada celda de una matriz de cadenas puede contener una cadena de una longitud de hasta 255 caracteres, igual que las variables tipo cadena típicas. Las matrices tipo cadena pueden ser uni o bidimensionales. De hecho, muchas versiones del BASIC, incluyendo el Applesoft, aceptan matrices de cadena y numéricas con más de dos dimensiones.

Ejemplo El siguiente programa asigna cuatro cadenas a una matriz tipo cadena unidimensional:

```
100 DATA "DALLAS", "CHICAGO"
105 DATA "MIAMI", "BRONX"
120 READ C$(1),C$(2),C$(3),C$(4)
130 LET C$(2)=C$(4)
140 IF C$(2)<>C$(1)THEN
    LET C$(2)=C$(1)
150 END
```

	C$(1)	C$(2)	C$(3)	C$(4)
	DALLAS	CHICAGO	MIAMI	BRONX
	DALLAS	BRONX	MIAMI	BRONX
	DALLAS	DALLAS	MIAMI	BRONX
	DALLAS	DALLAS	MIAMI	BRONX

8.6

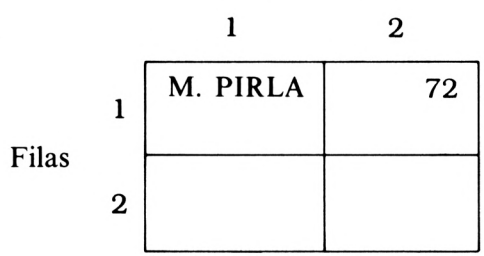
Ejercicios

1. Escriba un programa que ordene la anterior cadena que contiene las cuatro ciudades. También, que el programa imprima la matriz ordenada.

2. Escriba un programa general que ordene una matriz de cadena unidimensional arbitraria.

Ejemplo El siguiente es un ejemplo de matriz de cadena bidimensional que contiene nombres y edades de personas.

```
100 DIM A$(2,2)
110 DATA "M. PIRLA","72"
120 DATA "K. SAENZ","68"
130 READ A$(1,1), A$(1,2)
```



```
140 READ A$(2,1), A$(2,2)
```

Filas	Columnas 1	Columnas 2
1	M. PIRLA	72
2	K. SAENZ	68

```
150 END
```

Ejemplo La siguiente lista es el directorio telefónico personal de Manuel:

Garaje de Pedro	234-5456
Guillermo Negral	555-5655
Enrique Osuna	234-5458
Juana Tornado	874-5673
Panadería de Juani	871-2134
Café de Antonio	456-7676

El que sigue es un programa que

1) almacena la lista en la matriz tipo cadena K$,
2) busca un nombre requerido y visualiza el nombre y el número de teléfono.

```
100 DIM K$(6,2)
110 DATA "GARAJE DE PEDRO","234-5456","GUILLERMO
    NEGRAL","555-5655"
120 DATA "ENRIQUE OSUNA","234-5458","JUANA
    TORNADO","874-5673"
130 DATA "PANADERIA DE JUANI","871-2134","CAFE DE
    ANTONIO","456-7676"
150 REM LOS NOMBRES VAN EN LA COLUMNA 1 DE LA MATRIZ
160 REM LOS NUMEROS DE TELEFONO VAN EN LA COLUMNA 2
170 FOR I=1 TO 6
180 READ K$(I,1),K$(I,2)
190 NEXT I
200 INPUT "POR FAVOR, INTRODUZCA EL NOMBRE
    COMPLETO: ";N$
205 LET X=0
210 FOR I=1 TO 6
220 IF N$=K$(I,1) THEN PRINT K$(I,1),K$(I,2) :LET X=1
```

```
240 NEXT I
250 IF X=0 THEN PRINT "LO SIENTO, EL NOMBRE NO ESTA
    EN EL DIRECTORIO."
270 END
```

```
]RUN

POR FAVOR INTRODUZCA EL NOMBRE COMPLETO:
GUILLERMO NEGRAL
GUILLERMO NEGRAL          555-5655
```

8.6

Ejercicios (cont.)

3. Modifique el programa anterior de tal manera que sólo necesite ser introducida por el usuario una *clave de búsqueda.* La clave de búsqueda es una o más letras del principio del nombre. El ordenador debe imprimir el nombre completo y el número de teléfono de cualquiera de los nombres que coincidan con la clave. (NOTA: sólo necesita ser cambiada la línea 220.)

Es posible ordenar las cadenas de la misma manera que se ordenan datos numéricos.

Ejemplo Daniel Martos desea escribir un programa que ordene alguno de los importantes cursos académicos ofrecidos por el departamento de Humanidades de la Universidad.

```
RUDIMENTOS DEL BALONCESTO
HISTORIA DEL ROCK AND ROLL
ASTROLOGIA
HISTORIA DEL TEBEO
ESCRITORES MENORES DESPUES DE SHAKESPEARE
```

El siguiente programa ordenará e imprimirá la lista anterior en orden alfabético:

```
100 REM ORDEN ALFABETICO ASCENDENTE
105 DATA 5
110 DATA "RUDIMENTOS DEL BALONCESTO"
115 DATA "HISTORIA DEL ROCK AND ROLL"
120 DATA "ASTROLOGIA","HISTORIA DEL TEBEO"
130 DATA "ESCRITORES MENORES DESPUES DE SHAKESPEARE"
135 READ N
140 DIM R$(N)
150 FOR I=1 TO N
160 READ R$(I)
```

```
170 NEXT I
180 FOR U=1 TO N-1
190 FOR T=1 TO N-1
200 IF R$(T)>R$(T+1) THEN LET L$=R$(T) :LET R$(T)=R$(T+1) :
    LET R$(T+1)=L$
210 NEXT T
220 NEXT U
230 FOR I=1 TO N
240 PRINT R$(I)
250 NEXT I
260 END
```

```
]RUN

ASTROLOGIA
ESCRITORES MENORES DESPUES DE SHAKESPEARE
HISTORIA DEL ROCK AND ROLL
HISTORIA DEL TEBEO
RUDIMENTOS DEL BALONCESTO
```

APLICACION DE LA ELECCION ALEATORIA

El Club de Magia de Madrid tiene cinco miembros voluntarios para servir en el comité de "Limpieza". Ya que hay sólo tres plazas libres, el siguiente programa BASIC selecciona tres miembros al azar:

```
100 REM ELECCION ALEATORIA DE TRES MIEMBROS DE
110 REM UN GRUPO DE CINCO, SIN REPETICION.
120 DATA "JUAN","GUILLE","MARI","JAVI","FELIPE"
150 DIM P$(5)
160 FOR J=1 TO 5
170 READ P$(J)
180 NEXT J
185 FOR I=1 TO 3
190 LET L=INT(5*RND(1))+1
210 IF P$(L)="ELEGIDO YA" THEN GOTO 190
220 PRINT P$(L)
230 LET P$(L)="ELEGIDO YA"
240 NEXT I
250 END
```

```
]RUN
JAVI
FELIPE
MARI
```

Ejercicios (cont.)

4. El señor Masó prepara un partido de fútbol con catorce jugadores en la alineación inicial. Escriba un programa que seleccione su alineación inicial al azar. Invente sus propios nombres de jugadores.

APLICACION

La compañía Limón de coches de alquiler desea introducir en el ordenador la siguiente información acerca de cada uno de sus clientes:

1) Nombre del cliente.
2) Domicilio del cliente.
3) Número de teléfono del cliente.
4) Número de matrícula del automóvil.
5) Depósito del cliente.

El siguiente módulo de asignación cumplirá esta tarea:

```
100 INPUT "INTRODUZCA EL NUMERO DE CLIENTES: ";K
110 DIM R$(K,5)
120 FOR I=1 TO K
130 INPUT "INTRODUZCA EL NOMBRE DEL CLIENTE: ";R$(I,1)
140 INPUT "INTRODUZCA EL DOMICILIO DEL CLIENTE: ";R$(I,2)
150 INPUT "INTRODUZCA EL TELEFONO DEL CLIENTE: ";R$(I,3)
160 INPUT "INTRODUZCA LA MATRICULA DEL AUTOMOVIL: "
    R$(I,4)
170 INPUT "INTRODUZCA EL DEPOSITO DEL CLIENTE: ";R$(I,5)
175 PRINT
180 NEXT I
```

```
]RUN

INTRODUZCA EL NUMERO DE CLIENTES 2
INTRODUZCA EL NOMBRE DEL CLIENTE: CARLOS RIVAS
INTRODUZCA EL DOMICILIO DEL CLIENTE: OLMO 110
INTRODUZCA EL TELEFONO DEL CLIENTE: 760 - 1111
INTRODUZCA LA MATRICULA DEL AUTOMOVIL: 657.453
INTRODUZCA EL DEPOSITO DEL CLIENTE: 500
INTRODUZCA EL NOMBRE DEL CLIENTE: GUILLERMO SANZ
INTRODUZCA EL DOMICILIO DEL CLIENTE: ATOCHA 28
INTRODUZCA EL TELEFONO DEL CLIENTE: 345 - 4566
INTRODUZCA LA MATRICULA DEL AUTOMOVIL: 554.366
INTRODUZCA EL DEPOSITO DEL CLIENTE: 345
```

Este programa puede ser desarrollado para seleccionar un cliente por el nombre, de tal manera que pueda imprimirse la ficha de cualquier cliente en la pantalla.

```
200 INPUT "INTRODUZCA EL NOMBRE DESEADO: ";N$
205 LET G=0
210 FOR I=1 TO K
220 IF N$=R$(I,1) THEN LET G=I
230 NEXT I
240 IF G=0 THEN PRINT "LO SIENTO, EL NOMBRE NO ESTA." :
      GOTO 260
250 PRINT R$(G,1), R$(G,2), R$(G,3), R$(G,4), R$(G,5)
260 END
```

```
(continuación del anterior)

INTRODUZCA EL NOMBRE DESEADO: GUILLERMO SANZ
GUILLERMO SANZ        ATOCHA 28
345 - 4566            554.366          345
```

8.6

Ejercicios (cont.)

5. Desarrolle el programa anterior para calcular el depósito total hecho por los clientes.

6. Modifique el programa anterior de tal manera que uno pueda obtener los nombres de los clientes que hacen depósitos entre 10000 y 20000 pesetas.

7. Modifique el programa anterior de tal manera que introduciendo una matrícula, se visualice la ficha del cliente en la pantalla.

8.7

Proyectos

1. Un cuadrado mágico es una matriz cuadrada de números, tal que todas las filas, columnas y diagonales sumen lo mismo.

Ejemplo

8	1	6	1	15	14	4
3	5	7	12	6	7	9
4	9	2	8	10	11	5
			13	3	2	16

Escriba un programa BASIC para comprobar si una matriz es un cuadrado mágico. Los números del cuadrado deben estar en sentencias DATA en el programa.

2. Los números hexadecimales tienen muchas aplicaciones en informática. Igual que los números binarios son reducidos a dos dígitos, el 0 y el 1, los números hexadecimales están hechos de 16 dígitos y letras 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F. Las letras de la A a la F sustituyen a los números decimales del 10 al 15. Los valores-lugar hexadecimales de derecha a izquierda son, 1, 16, 256 (=16*16), 4096 (=16*16*16), etc.

Ejemplos 1F (hex)=31 (decimal)

A10 (hex)=2576 (decimal)

a) Escriba un programa que convierta un número hexadecimal arbitrario en su decimal equivalente.

b) Escriba un programa que convierta un número entero decimal arbitrario en su hexadecimal equivalente. (Véase el programa de conversión de números decimales en binarios de la sección 8.4.)

3. Considere las sentencias DATA siguientes:

```
DATA "AS","DOS","TRES","CUATRO"
DATA "CINCO","SEIS","SIETE"
DATA "OCHO","NUEVE","DIEZ"
DATA "SOTA","CABALLO","REY"
DATA "OROS","COPAS"
DATA "ESPADAS","BASTOS"
```

Utilizando estas sentencias, escriba un programa que:

a) Almacene las cincuenta y dos cartas de una baraja ordinaria en una matriz A bidimensional de cincuenta y dos filas por dos columnas. La primera columna de A contiene el valor de la carta, la segunda contiene el palo.

b) Imprima cuatro manos de cinco cartas, en donde las veinte cartas sean elegidas arbitrariamente sin repetición.

4. La tabla siguiente es el total de dólares de ventas para cuatro vendedores que vendieron dos modelos de automóvil:

Vendedor	Modelo A	Modelo B	Total
Julián	345,567	567,543	
Guillermo	897,444	908,661	
Felipe	123,199	234,567	
Antonio	345,111	222,333	
Total			

Escriba un programa BASIC que imprima esta tabla con las sumas de cada una de las columnas y filas, clasificadas apropiadamente.

5. LOS SIGNOS DEL ZODIACO

Dados el día y el mes del nacimiento, escriba un programa que determine el correspondiente signo del zodiaco. La siguiente información da la fecha del nacimiento y el signo del zodíaco.

Signo	Período
Capricornio	dic 22 - ene 19
Acuario	ene 20 - feb 18
Piscis	feb 19 - mar 20
Aries	mar 21 - abr 19
Tauro	abr 20 - may 20
Géminis	may 21 - jun 20
Cáncer	jun 21 - jul 22
Leo	jul 23 - ago 22
Virgo	ago 23 - sep 22
Libra	sep 23 - oct 22
Escorpio	oct 23 - nov 21
Sagitario	nov 22 - dic 21

Este programa debe determinar el signo del zodíaco cuando el usuario introduzca el día y mes.

PRACTICA 8.1

Matrices unidimensionales

Introduzca la orden: NEW

Una variable con índice (variable indexada) consta de un nombre de variable seguido de un índice encerrado entre paréntesis. Por ejemplo, A(1) es una

variable indexada, pero A1 no. Un conjunto de variables indexadas con el mismo nombre de variable se llama *matriz.*

Introduzca el programa:

```
10 FOR K=0 TO 10
20 READ Z$(K)
30 LET X(K)=K+1
40 NEXT K
50 PRINT "K","Z$(K)","X(K)"
60 FOR K=0 TO 10
70 PRINT K,Z$(K),X(K)
80 NEXT K
90 END
100 DATA A,B,C,D,E,F,G,H,I,J,K
```

Introduzcà la orden: RUN

Obsérvese que el índice es una variable. Cuando K=0, Z$(K) es Z$(0) y X(K) es X(0). Cuando K=1, Z$(K) es Z$(1) y X(K) es X(1).

Estudiando la respuesta del programa, Z$(0) contiene el número 1.

¿Cuáles son los contenidos de Z$(K) e Y(K) cuando J=4? _____

La cadena E se almacena en Z$(4) y el número 5 se almacena en X(4). El Applesoft BASIC admite automáticamente 11 posiciones en la memoria (de 0 a 10) por cada matriz unidimensional.

Introduzca las sentencias:

```
10 FOR K=0 TO 15
60 FOR K=0 TO 15
```

Introduzca la orden: LIST

Ya que a K le serán asignados enteros del 0 al 15, se necesitarán para ambas matrices, Z$(K) y X(K), dieciséis localizaciones.

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

La causa del error es la falta de posiciones en la memoria para la matriz. Esto es, el ordenador no tiene reservadas posiciones en la memoria para Z$(11) hasta Z$(15) y X(11) hasta X(15). Recuerde, el Applesoft BASIC reserva automáticamente 11 posiciones en la memoria (índices del 0 al 10).

Cuando se necesitan más posiciones en la memoria para la matriz, se usa la sentencia DIM.

Introduzca la sentencia: 5 DIM Z$(15),X(15)

Introduzca la orden: LIST

Nótese que la sentencia DIM es la primera sentencia del programa. Todas las sentencias DIM deben colocarse al principio de un programa antes de cualquier sentencia que use matrices. Esto asegurará que estén apropiadamente dimensionadas todas las matrices del programa.

Introduzca la orden: RUN

Explique lo que hace el ordenador cuando ejecuta la línea número 5. _____

_____

_____

_____

Cuando el ordenador ejecuta la línea número 5, reserva dieciséis posiciones de almacenamiento en la memoria para las matrices con nombres de variables Z$(0) a Z$(15) y X(0) a X(15).

El índice debe ser un número no negativo.

Introduzca la orden: NEW

Introduzca las sentencias:
```
10 LET A(-2)=5
20 PRINT A(-2)
30 END
```

Introduzca la orden: RUN

¿Cuál es la respuesta del ordenador? _____

Explique por qué ocurre el error _____

_____

Recuerde, los índices deben ser mayores o iguales a cero (0).

Si el índice no es un número entero, BASIC lo redondeará para convertirlo en uno entero.

Introduzca la orden: NEW

Introduzca las sentencias:
```
10 LET A(1.8)=5
20 PRINT A(1)
```

Introduzca las órdenes: LIST
RUN

¿Cuál es la respuesta del ordenador? _____

Explique por qué la respuesta es 5. _____

_____

_____

El número 1.8 es redondeado en la línea 10 resultando el entero 1. Así, a la variable indexada A(1) le es asignado el valor 5.

PRACTICA 8.2

Las matrices bidimensionales

Introduzca la orden: NEW

Una tabla rectangular de valores que conste de filas y columnas se llama "matriz bidimensional". La sentencia DIM se usa para crear una matriz bidimensional de variables indexadas numéricas o de cadena.

Introduzca la sentencia: 10 DIM X$(2,3)

Los dos argumentos de dentro del paréntesis deben estar separados por una coma. Pueden ser números no negativos, variables numéricas, o expresiones aritméticas que generan números no negativos.

El primer argumento, 2, es el valor indexado más alto de las filas. El segundo argumento, 3, es el valor indexado más alto de las columnas de la matriz de orden 2. El valor más bajo de filas y columnas es el 0.

Así, la ejecución de la sentencia 10 DIM X$(2,3) crea una matriz bidimensional con tres filas (0, 1 y 2) y cuatro columnas (0, 1, 2 y 3). En la tabla siguiente se muestran doce localizaciones de almacenamiento reservadas en la memoria con los nombres de variables:

	Columna 0	Columna 1	Columna 2	Columna 3
Fila 0	X$(0,0)	X$(0,1)	X$(0,2)	X$(0,3)
Fila 1	X$(1,0)	X$(1,1)	X$(1,2)	X$(1,3)
Fila 2	X$(2,0)	X$(2,1)	X$(2,2)	X$(2,3)

Introduzca la sentencia: 10 DIM X$(1,2)

Cuando el ordenador ejecute esta sentencia, ¿cuántas posiciones de almacenamiento reservará? _____

El ordenador reservará seis posiciones de almacenamiento.

¿Cuáles son los nombres de las variables de la matriz? _____

_____

Los nombres de las variables de la matriz son X$(0,0), X$(0,1), X$(0,2), X$(1,0), X$(1,1), X$(1,2).

Introduzca las sentencias:
```
20 LET X$(2,2)="HOLA"
30 END
```

Introduzca las órdenes:
```
LIST
RUN
```

¿Cuál es la respuesta del ordenador? _____

Explique por qué se produce el error. _____

_____

El ordenador no reserva una posición de almacenamiento para el nombre de la variable X$(2,2). La celda numerada (2,2) está fuera del rango de la matriz dimensionada.

Introduzca la orden: NEW

Introduzca el programa:
```
10 DIM A(1,2)
20 FOR J=0 TO 2
30 READ A(0,J)
40 NEXT J
50 FOR J=0 TO 2
60 READ A(1,J)
70 NEXT J
80 PRINT "A(0,0)=";A(0,0)
90 PRINT "A(0,1)=";A(0,1)
100 PRINT "A(0,2)=";A(0,2)
110 PRINT "A(1,0)=";A(1,0)
120 PRINT "A(1,1)=";A(1,1)
130 PRINT "A(1,2)=";A(1,2)
140 DATA 10,20,30,40,50,60
150 END
```

Introduzca las órdenes: LIST
RUN

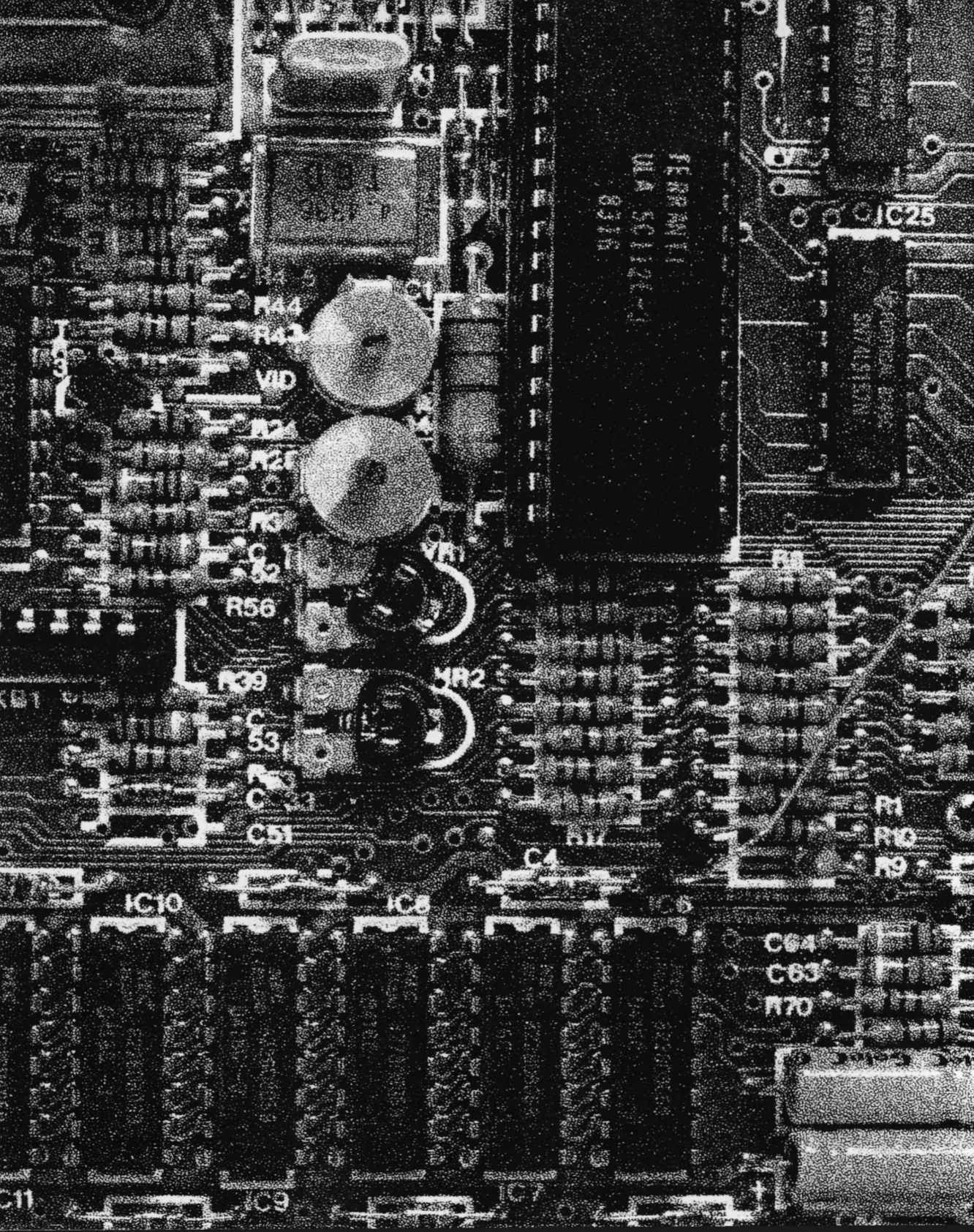
Normalmente se utilizan los lazos anidados para asignar datos a una matriz bidimensional, así como para imprimir en la pantalla los datos de una matriz bidimensional. El programa siguiente es una mejora sobre el programa anterior.

Introduzca la orden: NEW

Introduzca el programa:

```
 10 DIM A(1,2)
 20 FOR I=0 TO 1
 30 FOR J=0 TO 2
 40 READ A(I,J)
 50 NEXT J
 60 NEXT I
 70 FOR I=0 TO 1
 80 FOR J=0 TO 2
 90 PRINT "A(";I",";J")=";A(I,J)
100 NEXT J
110 NEXT I
120 DATA 10,20,30,40,50,60
```

Introduzca las órdenes: LIST
RUN


IC25
R44
VID
R56
R39
C51
C4
R1
R9
IC10
IC8
C64
C63
R70
IC9
IC7

9
Más sobre cadenas

9.0
Introducción

Ahora trataremos algunas importantes aplicaciones relacionadas con el tratamiento de datos en forma de cadena. El hecho de unir dos o más cadenas para formar una única es fundamental para todo lo relacionado con este capítulo. A este proceso se le llama *concatenación.*

9.1
Concatenación

La concatenación es una operación con cadenas similar a la adición con números. Concatenar dos cadenas es unirlas para formar una única. El símbolo de la concatenación es el signo más (+). Se sitúa entre las dos cadenas que hay que unir.

Ejemplo

```
10 LET A$ = "JUAN"
20 LET B$ = "JOSE"
30 LET C$ = A$ + B$
40 END
```

A$	B$	C$
JUAN		
JUAN	JOSE	
JUAN	JOSE	JUANJOSE

Es en la línea 30 donde tiene lugar la concatenación. Las variables tipo cadena están en el orden en que deben ser concatenadas, y separadas por un signo más, que es el operador concatenación. A C$ se le asigna el resultado de la operación.

Ejemplo El siguiente programa utiliza la concatenación para generar el nombre de las cincuenta y dos cartas de la baraja francesa:

```
100 DATA "AS","DOS","TRES"
110 DATA "CUATRO","CINCO","SEIS"
120 DATA "SIETE","OCHO","NUEVE"
130 DATA "DIEZ","JACK","REINA"
140 DATA "REY"
160 DATA "CORAZONES","DIAMANTES"
170 DATA "TREBOLES","PICAS"
180 DIM A$(13), B$(4), C$(52)
190 FOR I=1 TO 13
200 READ A$(I)
210 NEXT I
220 FOR I=1 TO 4
230 READ B$(I)
240 NEXT I
245 LET K=0
250 FOR I=1 TO 13
260 FOR J=1 TO 4
265 LET K=K+1
270 LET C$(K)=A$(I)+" DE"+B$(J)
280 NEXT J
290 NEXT I
300 FOR K=1 TO 52
310 PRINT C$(K)
320 NEXT K
330 END
```

```
]RUN
AS DE CORAZONES
AS DE DIAMANTES
AS DE TREBOLES
AS DE PICAS
DOS DE CORAZONES
DOS DE DIAMANTES
DOS DE TREBOLES
DOS DE PICAS
      .
      .
      .
REY DE CORAZONES
REY DE DIAMANTES
REY DE TREBOLES
REY DE PICAS
```

9.1

Ejercicios

1. Si en lugar de lo escrito en la línea 270 se hubiera escrito

   LET C$(K) = A$(I) + B$(J)

   ¿qué diferencia se produciría?

2. Los señores Pérez acaban de tener su primer hijo, y es niña. Han decidido que su primer nombre va a ser Mercedes, Cristina, Amelia o Carlota y que su segundo nombre va a ser Ana, Daniela, María o Manuela. Ayúdeles a decidir el nombre completo de su hija escribiendo un programa que genere todas las combinaciones posibles (dieciséis) acompañadas del apellido. Utilice la concatenación para generar los nombres completos.

3. Escriba un programa que traduzca cualquier palabra en otra mediante las reglas siguientes: quitar la primera letra, situarla al final de la palabra, y añadirle AY. Por ejemplo, BASIC se convierte en ASICBAY. Supóngase que todas las palabras se traducen de la misma forma.

4. Escriba un programa que tomando un nombre y un apellido de personas los invierta. Por ejemplo, Juan Pérez será escrito como Pérez Juan.

9.2

Escritura de números en pesetas y céntimos

Hacer que el ordenador escriba los números en pesetas y céntimos no es tan sencillo como parece. Números como el 3.5 y el 23 deben ser transformados en 3.50 y 23.00. Otros números, como 2.587 por ejemplo,

debe ser redondeado a 2.59. He aquí un esquema de lo que debe hacer el programa:

1) Mover el punto decimal dos espacios a la derecha y redondear al número entero más cercano. Por ejemplo, 2.587 se convierte en 259.

2) Pasar el resultado del paso 1 a cadena y concatenarlo con "00" por su izquierda. Continuando con el ejemplo anterior, 259 se convierte en 00259.

3) Asignar los dos dígitos más a la derecha a R$. Por ejemplo, poner 59 (de 00259) en R$.

4) Asignar los dígitos restantes a L$. En nuestro ejemplo, 002 en L$.

5) Quitar los ceros a la izquierda de L$ pasándolo a número (usando la función VAL) y nuevamente a cadena (con STR$). En nuestro ejemplo, VAL (L$) es 2, que cambiado de nuevo a cadena se le asigna AL$.

6) Concatenar L$ y R$ con un punto en medio, y ya está. El resultado es 2.59.

He aquí un programa que realiza la función:

```
10 INPUT "CUAL ES EL NUMERO?";N
30 REM
40 REM PASO 1 - MOVER EL PUNTO DECIMAL DOS
   ESPACIOS Y REDONDEAR
50 LET N1 = INT(100*N + .5)
60 REM
70 REM PASO 2 - CAMBIAR N1 A CADENA Y UNIRLE "00"
80 LET N1$ = "00" + STR$(N1)
85 REM
90 REM PASO 3 - EXTRACCION DE LOS DOS DIGITOS MAS
   A LA DERECHA
100 LET R$ = RIGHT$(N1$,2)
105 REM
110 REM PASO 4 - EXTRACCION DE LOS DEMAS DIGITOS
120 LET L$ = LEFT$(N1$,LEN(N1$)-2)
125 REM
130 REM PASO 5 - QUITAR CEROS A LA IZQUIERDA EN L$
140 LET L$ = STR$(VAL(L$))
```

```
145 REM
150 REM PASO 6 - REUNIR TODAS LAS PARTES
160 LETN$ = L$ + "." + R$
165 REM
170 PRINT "HE AQUI SU NUMERO EN PESETAS Y CENTIMOS:";N$
180 END
```

```
]RUN
CUAL ES EL NUMERO? 2.587
HE AQUI SU NUMERO
2,59
```

9.2

Ejercicio

1. Combine los pasos 1 y 2 en una única sentencia LET. A continuación combine los pasos 3, 4, 5 y 6 en otra única sentencia LET. La ventaja de tener sólo dos LET en vez de seis es que sólo hay una variable intermedia, en lugar de tres.

9.3

Campos y registros

En muchas aplicaciones, caracteres situados en posiciones específicas dentro de una cadena pueden tener especial significado. Un grupo de caracteres con significado propio se llama *campo*. Una colección de campos se llama *registro*.

He aquí tres registros típicos de un listín de teléfonos:

```
SMITH       PAT         555-2414
JOHNSON     RICHARD     592-3425
SHAKESPEARWILLIAM       342-1449
12345678911111111122222222          (número guía)
         012345678901234567
```

Hay tres campos en cada registro. Estos son apellido, nombre, y teléfono. Fíjese que los mismos campos empiezan siempre en la misma posición en todos los registros. Una buena costumbre de programación es que registros que contienen el mismo tipo de información tengan campos de longitud fija. El número guía debajo de los registros nos permiten ver que las longitudes y posiciones de los campos son las siguientes:

Nombre del campo	Longitud	Posición
Apellido	10	1 - 10
Nombre	9	11 - 19
Teléfono	8	20 - 27

Es obvio que no todos los apellidos van a tener diez letras. Algunos tendrán más; otros menos. Aún así, el concepto de campo de longitud fija no requiere que todos los caracteres del campo estén ocupados. Para nombres cortos, los restantes caracteres del campo se rellenarán con espacios. Para nombres más largos, éste se truncará (véase Willian Shakespeare).

El siguiente programa acepta el apellido, nombre y teléfono y escribe un registro con los campos como los descritos arriba:

```
 10 REM CREACION DE UN REGISTRO CON TRES CAMPOS
 20 REM A B$ SE LE ASIGNAN 10 ESPACIOS
 30 LET B$ = "          "
 40 REM
 50 REM ASIGNACION DE CAMPOS
 60 INPUT "APELLIDO ";A$
 70 INPUT "NOMBRE ";N$
 80 INPUT "TELEFONO ";T$
 90 REM
100 REM AJUSTAR LOS CAMPOS A LA LONGITUD CORRECTA
110 LET A$ = A$ + B$
120 LET A$ = LEFT$(A$,10)
130 LET N$ = N$ + B$
140 LET N$ = LEFT$(N$,9)
150 LET T$ = T$ + B$
160 LET T$ = LEFT$(T$,8)
170 REM
180 REM CONCATENADO DE TODOS LOS CAMPOS PARA
    CREAR EL REGISTRO
190 LET R$ = A$ + N$ + T$
200 PRINT
210 END
```

```
]RUN

APELLIDO: SMITH
NOMBRE: PAT
TELEFONO: 555-2414
SMITH      PAT      555-2414
```

Es interesante el modo en que cada campo se ajusta a su longitud. Una cadena de espacios tan larga, al menos, como el campo más largo, se concatena con cada ítem introducido. Entonces, utilizando la función LEFT$, recortamos la cadena al tamaño deseado.

La principal razón para insistir en los campos con longitud fija es debido a que luego seremos capaces, gracias a esto, de coger un registro arbitrario y saber exactamente dónde mirar para encontrar la información deseada. El ejemplo siguiente utiliza la idea de los campos de longitud fija para ordenar un listín telefónico por orden numérico ascendente.

Ejemplo

El siguiente programa acepta nombres y teléfonos de N personas y los ordena por el número de teléfono:

```
 10 REM MODULO DE ASIGNACION
 20 INPUT "CUANTAS PERSONAS? ";N
 30 DIM R$(N)
 40 LET B$ = "                    "
 50 FOR I=1 TO N
 60 PRINT
 70 INPUT "APELLIDO: ";A$
 80 INPUT "NOMBRE: ";N$
 90 INPUT "TELEFONO: ";T$
100 LET L$ = LEFT$(A$ + B$,10)
110 LET F$ = LEFT$(N$ + B$,9)
120 LET T$ = LEFT$(T$ + B$,8)
130 LET R$ = A$ + N$ + T$
140 NEXT I
150 REM MODULO DE CALCULO, ORDENACION DE
    TELEFONOS
160 FOR J=1 TO N-1
170 FOR K=1 TO N-1
180 REM EL CAMPO DEL TELEFONO EMPIEZA EN LA
    POSICION 20
190 IF MID$(R$(K),20,8) > MID$(R$(K + 1),20,8) THEN
    LET X$ = R$(K) : LET R$(K) = R$(K + 1) : LET R$(K + 1) = X$
200 NEXT K
210 NEXT J
220 REM MODULO DE SALIDA
230 PRINT
240 FOR I=1 TO N
250 PRINT R$(I)
260 NEXT I
270 END
```

```
]RUN

CUANTAS PERSONAS? 3
APELLIDO: SMITH
NOMBRE: PAT
TELEFONO: 555-2414
APELLIDO: JOHNSON
NOMBRE: RICHARD
TELEFONO: 592-3425
APELLIDO: SHAKESPEARE
NOMBRE: WILLIAM
TELEFONO: 342-1449

SHAKESPEARWILLIAM     342-1449
SMITH         PAT     555-2414
JOHNSON       RICHARD 592-3425
```

En el programa anterior se han almacenado N registros en la matriz R$ Dicho conjunto de registros dedicado a una lista específica de datos se llama *archivo,* cuyo tratamiento se realizará en el capítulo 11.

9.3 Ejercicios

1. A pesar de que todos los campos son almacenados en cadenas, algunos pueden representar información numérica tales como edad, peso, salario, notas medias, etc. Estos campos deben ser *ajustados a la derecha* Esto significa que la información debe estar totalmente situada a la derecha y el resto relleno con espacios. Esto se hace para que al realizar ordenaciones en un campo numérico los resultados sean correctos (si están ajustados a la izquierda la cadena 25 se considerará mayor que la 191 porque 2 viene después "alfabéticamente" que 1) Reescriba el programa anterior de tal manera que cada registro contenga un cuarto campo: el salario mensual de cada persona. Ajuste a la derecha el salario en un campo de longitud 5 y ordene las entradas por salarios (cuando introduzca los salarios mensuales, no ponga céntimos).

2. Escriba un programa que

   1) Acepte una entrada formada por un número arbitrario de números enteros (sin punto decimal).

   2) Escriba una lista ordenada, con un número por línea, sin ceros a la izquierda, y ajustados a la derecha.

Ejemplo

Entrada: 0342, 00013, 24, 92142, 1
Salida:
1
13
24
342
92142

3. Escriba un programa que

   1) Acepte como entrada un número arbitrario de palabras.

   2) Escriba las palabras en orden alfabético, una por línea, y ajustadas a la izquierda.

Ejemplo

Entrada: perro, zapato, bolso, peine, impermeable
Salida:
bolso
impermeable
peine
perro
zapato

4. Para el problema 3 escriba un programa de tal manera que la lista esté ajustada a la derecha.

9.4

Proyectos

1. EL MINIPROCESADOR DE TEXTO-I.

Escriba un programa BASIC que realice las siguientes funciones:

a) Cuando el usuario escriba una frase en el teclado el ordenador debe escribirla en pantalla.

b) El usuario podrá sustituir una palabra por otra, y la frase modificada debe aparecer en pantalla.

2. EL MINIPROCESADOR DE TEXTO-II.

Escriba un programa BASIC que realice las siguientes funciones:

a) Cuando el usuario escriba una frase en el teclado el ordenador debe escribirla en pantalla.

b) El usuario podrá insertar una palabra entre cualquier otras dos de la frase, y la frase modificada debe aparecer en pantalla.

3. EL MINIPROCESADOR DE TEXTO-III.

Los procesadores de texto típicos tienen la habilidad de tomar una carta tipo y "personalizarla". Hacen esto buscando en la carta la existencia de símbolos especiales, tales como *NOMBRE*, y reemplazando el símbolo por un nombre real. Realice un programa que haga esto.

4. La señora Otto desea alquilar un coche para conducir de Los Angeles a San Francisco (400 millas). No desea repostar gasolina en ruta. Además, el precio total del viaje no debe exceder de 200 dólares. Ayúdela a escribir un programa para elegir un coche o coches de la siguiente lista, que se ajuste a sus requerimientos. Suponga que el precio de la gasolina es de 1.50 dólares el galón:

Auto	Modelo	Millas por galón	Capacidad	Precio día	Precio milla
Hot Rod	Fast	18	16	20	.25
Gasser	Slow	23	10	24	.27
Stars	Speedy	29	20	29	.30
Fire	Classy	25	21	30	.22
Tevi	Sharp	17	15	22	.32
Caby	Tops	27	17	23	.18

5. Con la tabla anterior, escriba un programa que produzca dos tablas ordenadas, siendo las ordenaciones según los campos de millas por galón y precio día.

PRACTICA 9.1

Concatenación

Escriba la orden: NEW

Concatenar significa unir dos cadenas para formar una serie. Cuando es necesario unir dos cadenas, se utiliza la concatenación.

Escriba el programa:

```
10 LET A$ = "DOWN"
20 LET B$ = "TOWN"
30 LET C$ = A$ + B$
40 PRINT C$
50 END
```

Escriba la orden: RUN

¿Cuál es la respuesta del programa? _____

La concatenación tiene lugar cuando el ordenador ejecuta la línea 30. La cadena que hay en A$ se une a la que hay en B$ utilizando el operador +. La cadena concatenada se almacena en C$. Fíjese en el orden en que las dos cadenas están unidas. La ordenación de las variables en una concatenación es importante.

Escriba las sentencias:
```
50 LET D$ = B$ + A$
60 PRINT D$
70 END
```

Escriba la orden:
```
LIST
RUN
```

¿Cuál es la respuesta del programa? _____

_____

Explique por qué C$ *y* D$ *son distintos.* _____

_____

_____

_____

Ambas cadenas almacenadas en C$ y D$ fueron formadas por la concatenación de A$ y B$. Aún así, el orden de las variables en la línea 30 es distinto que en la línea 50.

Escriba las sentencias:
```
30 LET C$ = "LOW" + A$
50 LET D$ = B$ + "FOLKS"
```

Escriba las ordenes:
```
LIST
RUN
```

¿Cuál es la respuesta del programa? _____

_____

Se pueden concatenar tantas variables tipo cadena, cadenas o funciones que producen cadenas como se deseen, hasta el límite de 255 caracteres en la cadena resultante.

Escriba la orden: NEW

Escriba el programa:

```
10 DATA UNO,DOS,TRES,CUATRO,CINCO
20 FOR I=1 TO 5
30 READ N$
40 LET A$(I)="EL SIMBOLO DE "+N$+" ES "+STR$(I)
50 NEXT I
60 FOR K=1 TO 5
70 PRINT A$(K)
80 NEXT K
90 END
```

Escriba la orden: RUN

Explique la concatenación efectuada en la línea 40. _____

_____

_____

PRACTICA 9.2

Ajustado y rellenado

Escriba la orden: NEW

Todos los registros de una fila deben estar construidos idénticamente, esto es, las longitudes de los campos y las posiciones deben ser las mismas en cada registro. Dado que los datos almacenados en los mismos campos de distintos registros pueden tener longitudes distintas, es necesario el *rellenado* para cubrir las posiciones del campo que no son usadas. El siguiente programa muestra qué ocurre cuando no se rellenan los campos.

Escriba el programa:

```
10 HOME
20 FOR K=1 TO 3
30 INPUT "ESCRIBA UN NOMBRE: ";N$
40 INPUT "ESCRIBA UN TELEFONO: ";T$
50 LET A$(K)=N$+" "+T$
60 PRINT
70 NEXT K
80 FOR K=1 TO 3
90 PRINT A$(K)
110 NEXT K
120 END
```

Escriba la orden: RUN

Escriba los siguientes datos: JOE COOL
(213) 555-1000
SUZY Q
731-2222
BEATRIZ BENAVIDES
(805) 661-9999

¿Cuántos campos tiene cada registro? _____

Cada registro tiene dos campos. El primer campo contiene el nombre y el segundo el teléfono. Pero los campos no son consistentes en longitud o posición. No hay forma de saber, en cualquier registro, dónde acaba el nombre y dónde empieza el teléfono.

Para que el ordenador sea capaz de recuperar los datos almacenados en un registro, debe saber dónde está situado cada campo dentro del registro. Y esto es posible sólo si la longitud del campo no varía de registro. Cuando se diseña un registro, el programador debe primero determinar el número de campos y el máximo número de caracteres de cada campo.

En el programa anterior, ¿cuántos campos y cuántos caracteres son necesarios para cada campo?

_____

_____

Se necesitan dos campos. El número de caracteres en cada uno de ellos puede variar. Por ejemplo, Beatriz Benavides tiene 17 caracteres, pero quién es capaz de afirmar que éste es el mayor número de caracteres necesarios para un nombre. El diseño del registro debe ser tal que satisfaga la mayoría de los casos. Diseñemos nuestro registro de tal manera que el primer campo tenga 20 caracteres y el segundo 15. Las posiciones sin usar de cada campo se rellenan con espacios en blanco. A esto se le llama *rellenado.*

¿Cuántos caracteres tendrá cada registro de los anteriores? _____

Cada registro de los anteriores tendrá 35 caracteres.

El proceso de rellenado empieza por la asignación de un número de espacios igual a la longitud del mayor campo a una variable tipo cadena. En nuestro caso, el mayor campo tiene 20 caracteres.

Escriba la sentencia: 5 LET B$ = " " :REM (20 ESPACIOS)

Escriba la siguiente orden: LIST

Un campo se dice que está ajustado a la izquierda si el primer carácter de los datos está en la posición más a la izquierda del campo. Un campo se dice que está ajustado a la derecha si el último carácter de los datos está en la posición más a la derecha del campo.

Continuando con nuestro ejemplo, diseñaremos un registro de tal manera que el nombre esté ajustado a la izquierda en el primer campo y el teléfono ajustado a la derecha en el segundo campo.

Escriba las instrucciones:
```
45 LET F1$ = N$ + B$
50 LET F2$ = B$ + T$
```

Escriba la orden: LIST

Hemos concatenado espacios después de los datos ajustados a la izquierda y antes de los datos ajustados a la derecha.

Concatenar espacios después de los datos ajusta a la izquierda una variable tipo cadena. Concatenar espacios antes de los datos ajusta a la derecha una variable tipo cadena.

¿Qué número de línea justifica la izquierda de una variable tipo cadena y qué número de línea justifica la derecha de una variable tipo cadena?

_____

La línea número 45 concatena veinte espacios después de los datos almacenados en N$. Esto ajusta la izquierda de la variable tipo cadena F1$. En la línea número 50, se concatenan veinte espacios delante de los datos almacenados en T$. Por tanto, este número de línea ajusta la derecha de la variable tipo cadena F2$.

Escriba la sentencia:
```
55 LET A$(K) = LEFT$(F1$,20) + RIGHT$(F2$,15)
```

Escriba la orden: LIST

¿Qué campo está ajustado a la izquierda y qué campo está ajustado a la derecha en este programa? _____

_____

El primer campo (que contiene el nombre) está ajustado a la izquierda. El segundo campo (que contiene el número de teléfono) está ajustado a la derecha.

Escriba la orden: RUN

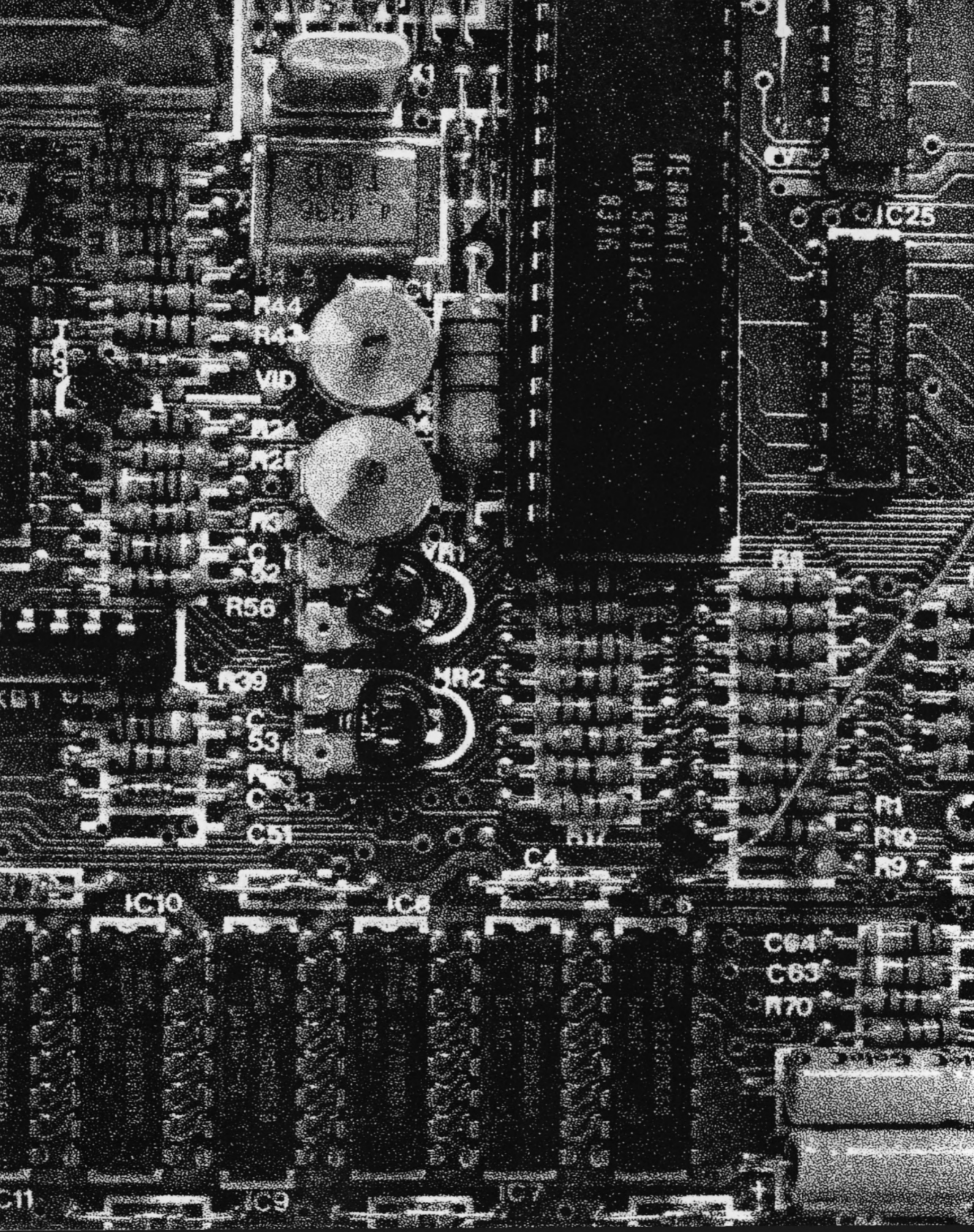
Escriba los mismos datos que antes.

¿Cuál es la razón más importante para el rellenado? _____

_____

_____

El rellenado hace que los campos que tienen similar tipo de información tengan el mismo número de caracteres en todos los registros. Por tanto, la localización de un dato dentro de un registro es más fácil, ya que todos los registros están estructurados de la misma manera.


IC25
R44
VID
R56
R39
C51
C4
R1
R9
IC10
IC8
C64
C63
R70
IC9
IC7

10 Programación estructurada

10.0

Introducción

La habilidad de programar consiste en algo más que en tener unos conocimientos de la sintaxis de un lenguaje de programación; se requiere que el programador sea capaz de escribir *algoritmos* o procesos cortos que resuelvan fácilmente los problemas planteados. A continuación los algoritmos son unidos para formar módulos. Los módulos son "miniprogramas" que realizan tareas específicas de la resolución de un problema más general. La solución total de un problema se realiza mediante la unión de varios *módulos,* y a esto lo podemos llamar *programa.* En este capítulo trataremos de algoritmos, módulos y programas.

10.1

Algoritmos

Un algoritmo es una regla o procedimiento para resolver un problema concreto. Por ejemplo, un algoritmo bien conocido en aritmética es el de

la división. Siguiendo siempre las reglas que nos enseñaron para dividir, tenemos la certeza de que llegaremos a la solución correcta, entendamos o no cómo funcionan esas reglas. Aquí trataremos varios algoritmos que pueden ser útiles en programas BASIC. Algunos de estos algoritmos ya han sido comentados en capítulos anteriores, por lo que nos resultarán familiares.

Obsérvese que ninguno de los algoritmos escritos a continuación tienen número de línea. Se trata de colocar los algoritmos en los lugares adecuados de sus programas y asignarle en ese momento el número de línea que le corresponda.

ALGORITMO I: ENTRADA DE UN NUMERO ARBITRARIO DE VALORES EN UNA MATRIZ UNIDIMENSIONAL

```
INPUT "CUANTOS NUMEROS DESEA INTRODUCIR? ";P
DIM N(P)
FOR I=1 TO P
INPUT "INTRODUZCA UN NUMERO: ";N(I)
NEXT I
```

Ejemplo La señora Sáenz quiere introducir en el ordenador las notas finales de sus alumnos. El siguiente algoritmo realizará la tarea:

```
10 INPUT "CUANTOS ALUMNOS TIENE? ";P
20 DIM N(P)
30 FOR I=1 TO P
40 INPUT "INTRODUZCA UNA NOTA ";N(I)
50 NEXT I
```

Ejemplo El señor Castro desea introducir en el ordenador los títulos de los libros de su biblioteca. El siguiente algoritmo realizará la tarea:

```
100 INPUT "CUANTOS LIBROS TIENE? ";N
110 DIM B$(N)
120 FOR I=1 TO N
130 INPUT "INTRODUZCA EL TITULO DE UN LIBRO: ";B$(I)
140 NEXT I
```

ALGORITMO II: SUMA DE UNA CANTIDAD ARBITRARIA DE NUMEROS EN UNA MATRIZ

```
REM P ES EL NUMERO DE VALORES A SUMAR
REM S ES LA VARIABLE QUE CONTIENE LAS SUMAS PARCIALES
```

```
REM N( ) ES LA MATRIZ QUE CONTIENE LOS NUMEROS
LET S=0
FOR I=1 TO P
LET S=S+N(I)
NEXT I
```

Ejemplo Marina vendió un número determinado de coches el último mes en la Lucky Auto Company. El siguiente programa sumará el total de sus comisiones:

```
100 INPUT "CUANTOS COCHES HA VENDIDO? ";P
110 DIM N(P)
120 FOR I=1 TO P
125 PRINT "INTRODUZCA LA COMISION POR LA VENTA DEL
    COCHE # ";I
130 INPUT N(I)
140 NEXT I
150 REM ALGORITMO II
160 LET S=0
170 FOR I=1 TO P
180 LET S=S+N(I)
190 NEXT I
200 PRINT "LA COMISION TOTAL DE MARINA ES ";S
210 END
```

ALGORITMO III: ENTRADA DE NUMEROS A UNA MATRIZ BIDIMENSIONAL

```
REM A( , ) ES UNA MATRIZ BIDIMENSIONAL
REM N ES EL NUMERO DE FILAS
REM M ES EL NUMERO DE COLUMNAS
INPUT "CUANTAS FILAS TIENE LA MATRIZ? ";N
INPUT "CUANTAS COLUMNAS TIENE LA MATRIZ? ";M
DIM A(N,M)
FOR I=1 TO N
FOR J=1 TO M
INPUT A(I,J)
NEXT J
NEXT I
```

Ejemplo El señor Luengo desea almacenar en el ordenador las notas puestas a sus N alumnos en los M problemas de un examen:

```
100 INPUT "CUANTOS ALUMNOS TIENE? ";N
110 INPUT "CUANTOS PROBLEMAS HAY POR EXAMEN? ";M
120 DIM A(N,M)
130 FOR I=1 TO N
140 PRINT "ESTUDIANTE # ";I
150 FOR J=1 TO M
160 PRINT "INTRODUZCA LA CALIFICACION DEL
    PROBLEMA # ";M
170 INPUT A(I,J)
180 NEXT J
190 NEXT I
```

10.1 Ejercicios

1. Escriba un algoritmo que introduzca cadenas en una matriz bidimensional.

ALGORITMO IV: SUMA DE LOS NUMEROS DE CADA UNA DE LAS FILAS DE UNA MATRIZ BIDIMENSIONAL

```
REM A( , ) ES LA MATRIZ
REM N ES EL NUMERO DE FILAS
REM M ES EL NUMERO DE COLUMNAS
REM S( )ES LA MATRIZ QUE CONTIENE LAS SUMAS PARCIALES
FOR I=1 TO N
LET S(I)=0
FOR J=1 TO M
LET S(I)=S(I)+A(I,J)
NEXT J
NEXT I
```

Ejemplo El señor Luengo desea sumar las calificaciones que cada alumno ha obtenido en los problemas. Supongamos que las siguientes líneas son la continuación del programa anterior:

```
175 DIM S(N)
180 FOR I=1 TO N
190 LET S(I)=0
195 FOR J=1 TO M
200 LET S(I)=S(I)+A(I,J)
210 NEXT J
220 PRINT "CALIFICACIONES POR ESTUDIANTE #";I;" ES ";S(I)
230 NEXT I
```

Ejercicios (cont.) 2. Escriba un algoritmo que sume los números de cada columna.

ALGORITMO V: ENTRADA DE UN NUMERO ARBITRARIO DE REGISTROS CON LOS CAMPOS AJUSTADOS POR LA IZQUIERDA

```
REM A1 ES LA LONGITUD DEL CAMPO F1$
REM A2 ES LA LONGITUD DEL CAMPO F2$, etc...
REM B$ ES UNA CADENA DE ESPACIOS TAN LARGA COMO
      EL MAS LARGO DE LOS CAMPOS
LET B$ = "                    "
INPUT "CUANTOS REGISTROS? ";Q
DIM R$(Q)
FOR I=1 TO Q
INPUT "INTRODUZCA CAMPO NUMERO 1: ";F1$
INPUT "INTRODUZCA CAMPO NUMERO 2: ";F2$
   .
   .
  etc.
   :
LET F1$ = LEFT$(F1$ + B$,A1)
LET F2$ = LEFT$(F2$ + B$,A2)
   :
  etc.
   :
LET R$(I) = F1$ + F2$ + ...etc...
NEXT I
```

Ejemplo La Yellow Stone Taxi Company quiere introducir la siguiente información de cada taxi en servicio en el ordenador:

1) Nombre del conductor.
2) Número de serie de taxi.
3) Años que lleva el taxi en servicio.
4) Distancia total viajada en un día.
5) Beneficios obtenidos.

El siguiente programa utiliza el algoritmo V para realizar esta tarea:

```
100 INPUT "CUANTOS TAXIS HAY EN SERVICIO? ";Q
110 DIM R$(Q)
```

```
120 FOR I=1 TO Q
130 LET B$ ="                                    "
140 INPUT "NOMBRE DEL CONDUCTOR: ";F1$
150 INPUT "NUMERO DE SERIE DEL TAXI: ";F2$
160 INPUT "AÑOS QUE LLEVA EL TAXI EN SERVICIO: ";F3$
170 INPUT "DISTANCIA TOTAL VIAJADA EN UN DIA: ";F4$
180 INPUT "BENEFICIOS OBTENIDOS: ";F5$
190 LET F1$ = LEFT$(F1$ + B$,25)
200 LET F2$ = LEFT$(F2$ + B$,10)
210 LET F3$ = LEFT$(F3$ + B$,2)
220 LET F4$ = LEFT$(F4$ + B$,4)
230 LET F5$ = LEFT$(F5$ + B$,4)
240 LET R$(I) = F1$ + F2$ + F3$ + F4$ + F5$
250 NEXT I
```

10.1

Ejercicios (cont.)

3. Empezando a continuación de la línea 250 del programa anterior, ampliarlo de manera que calcule la distancia total viajada y el dinero total recaudado por todos los taxis. No cambie nada anterior a la línea 250.

4. La Snappy Shoe Co. desea tener almacenada en su ordenador una lista con las direcciones de sus clientes. Se desea almacenar los siguientes datos de cada cliente:

   1) Apellido.
   2) Nombre.
   3) Domicilio.
   4) Población.
   5) Distrito postal.
   6) Número de teléfono.

   Realice un programa BASIC que cubra las especificaciones anteriores.

5. Modifique el algoritmo V de tal manera que los campos se ajusten por la derecha.

ALGORITMO VI: ORDENACION ASCENDENTE

```
REM N ES EL NUMERO DE ITEMS A SER ORDENADOS
FOR K=1 TO N-1
FOR J=1 TO N-K
```

```
IF A(J)>A(J+1) THEN LET L=A(J) :LET A(J)
      =A(J+1) :LET A(J+1)=L
NEXT J
NEXT K
```

ALGORITMO VIa: VERSION ALTERNATIVA DEL ALGORITMO DE ORDENACION ASCENDENTE

```
FOR K=1 TO N-1
FOR J=1 TO N-K
IF A(J)<=A(J+1) THEN GOTO línea 1
LET L=A(J)
LET A(J)=A(J+1)
LET A(J+1)=L
Línea 1 NEXT J
NEXT K
```

10.1

Ejercicios (cont.)

6. Escriba un algoritmo de ordenación descendente.

El campo sobre el que se ordena un grupo de registros se llama campo clave.

ALGORITMO VII: ORDENACION ASCENDENTE DE REGISTROS SOBRE UN CAMPO CLAVE

```
REM F ES LA LONGITUD DEL REGISTRO CLAVE
REM A ES LA POSICION DEL COMIENZO DEL REGISTRO CLAVE
FOR K=1 TO N-1
FOR J=1 TO N-K
IF MID$(R$(J),A,F)>MID$(R$(J+1),A,F) THEN
   LET L$=R$(J) :LET R$(J)=R$(J+1) :
   LET R$(J+1)=L$
NEXT J
NEXT K
```

Ejemplo La siguiente modificación del programa de la Yellow Stone Taxi Co., permite al usuario elegir el campo sobre el cual realizar la ordenación:

```
260 PRINT "1. NOMBRE DEL CONDUCTOR"
270 PRINT "2. NUMERO DE SERIE DEL TAXI"
```

```
280 PRINT "3. AÑOS QUE LLEVA EL TAXI EN SERVICIO"
290 PRINT "4. DISTANCIA TOTAL VIAJADA EN EL DIA"
300 PRINT "5. BENEFICIOS OBTENIDOS"
310 PRINT "PULSE EL NUMERO DEL CAMPO RESPECTO
    AL CUAL QUIERE QUE SE REALICE LA
    ORDENACION ";P
320 IF P=1 THEN LET A=1 :LET F=25
330 IF P=2 THEN LET A=26 :LET F=10
340 IF P=3 THEN LET A=36 :LET F=2
350 IF P=4 THEN LET A=38 :LET F=4
360 IF P=5 THEN LET A=42 :LET F=4
370 FOR K=1 TO Q-1
380 FOR J=1 TO Q-K
390 IF MID$(R$(J),A,F)>MID$(R$(J+1),A,F) THEN
    LET L$=R$(J) :LET R$(J)=R$(J+1) :
    LET R$(J+1)=L$
400 NEXT J
410 NEXT K
```

Si el campo clave está situado en el extremo derecho o en el extremo izquierdo del registro, resulta más conveniente la utilización de las funciones RIGHT$ o LEFT$ en vez de la MID$.

Ejemplo Lo que sigue es una modificación del programa de la Yellow Stone Taxi Co., en la que la matriz se ordena siguiendo los valores de los beneficios obtenidos:

```
260 REM ORDENACION RESPECTO AL CAMPO SITUADO
    MAS A LA DERECHA
270 FOR K=1 TO Q-1
280 FOR J=1 TO Q-K
290 IF RIGHT$(R$(J),4)>RIGHT$(R$(J+1),4) THEN
    LET L$=R$(J) :LET R$(J)=R$(J+1) :
    LET R$(J+1)=L$
300 NEXT J
310 NEXT K
```

10.1

Ejercicios (cont.)

7. Realice una modificación similar al programa de la Yellow Stone Taxi Co., de tal manera que realice la ordenación atendiendo a los nombres. Utilice la función LEFT$.

8. Amplíe el programa de la Snappy Shoe Co. (ejercicio 10.1.4), de tal manera que éste pueda realizar una ordenación respecto a cualquier campo elegido por el usuario.

ALGORITMO VIII: BUSQUEDA DE UNA CADENA EN CUALQUIER COLUMNA DE UNA MATRIZ BIDIMENSIONAL

```
REM R$(N,M) ES LA MATRIZ DE CADENAS
REM S$ ES LA CADENA BUSCADA EN LA COLUMNA C
LET L=0
FOR I=1 TO N
IF S$=R$(I,C) THEN LET L=I :LET I=N
NEXT I
IF L=0 THEN (instrucción para el caso de que no exista S$)
```

Ejemplo El profesor Fermi tiene grabado en su ordenador las notas de cuatro ejercicios realizados por sus N alumnos. Cada fila de la matriz bidimensional de cadenas G$ contiene el apellido y el nombre (separados por un espacio) de los alumnos seguidos por las cuatro calificaciones. El siguiente programa asigna los datos a la matriz, pregunta al usuario el nombre del alumno, busca a la izquierda de la columna de la matriz el nombre del alumno, y calcula e imprime la media aritmética de las notas obtenidas:

```
100 INPUT "¿CUANTOS ESTUDIANTES? ";N
110 DIM G$(N,5)
120 REM ASIGNACION DE LOS DATOS A LA MATRIZ
130 FOR I=1 TO N
140 PRINT :PRINT "ALUMNO ";I
150 INPUT "INTRODUZCA APELLIDO Y NOMBRE ";G$(I,1)
160 FOR J=2 TO 5
170 PRINT "INTRODUZCA LA NOTA DEL EJERCICIO ";J-1;
180 INPUT " NOTA: ";G$(I,J)
190 NEXT J
200 NEXT I
210 REM ALGORITMO VIII
220 REM RECUPERACION DE LOS DATOS MEDIANTE LA
    BUSQUEDA POR NOMBRE
230 PRINT
```

```
240 INPUT "INTRODUZCA EL APELLIDO DEL ESTUDIANTE:
    ";L$
250 INPUT "INTRODUZCA EL NOMBRE DEL ESTUDIANTE: ";F$
260 LET S$=L$+" "+F$
270 LET L=0
280 FOR I=1 TO N
290 IF S$=G$(I,1) THEN LET L=I :LET I=N
300 NEXT I
310 IF L=0 THEN PRINT "NO EXISTE TAL NOMBRE" :GOTO 390
320 REM FIN DEL ALGORITMO VIII
330 REM CALCULO DE LA MEDIA DE LAS NOTAS
340 LET S=0
350 FOR K=2 TO 5
360 LET S=S+VAL(G$(L,K))
370 NEXT K
380 PRINT "LA NOTA MEDIA DE ";S$;" ES ";S/4
390 END
```

```
]RUN
CUANTOS ESTUDIANTES? 2

ALUMNO 1
INTRODUZCA APELLIDO Y NOMBRE: MARTIN NORMA
INTRODUZCA LA NOTA DEL EJERCICIO 1: 98
INTRODUZCA LA NOTA DEL EJERCICIO 2: 92
INTRODUZCA LA NOTA DEL EJERCICIO 3: 88
INTRODUZCA LA NOTA DEL EJERCICIO 4: 96

ALUMNO 2
INTRODUZCA APELLIDO Y NOMBRE: RUIZ MARISOL
INTRODUZCA LA NOTA DEL EJERCICIO 1: 49
INTRODUZCA LA NOTA DEL EJERCICIO 2: 58
INTRODUZCA LA NOTA DEL EJERCICIO 3: 62
INTRODUZCA LA NOTA DEL EJERCICIO 4: 55
INTRODUZCA EL APELLIDO DEL ESTUDIANTE: RUIZ
INTRODUZCA EL NOMBRE DEL ESTUDIANTE: MARISOL
LA NOTA MEDIA DE RUIZ MARISOL ES 56
```

10.1

Ejercicios (cont.)

9. Escriba un algoritmo que busque una matriz bidimensional a lo largo de cualquier fila.

El siguiente ejemplo utiliza el algoritmo VIII y el algoritmo que usted desarrolló en el ejercicio 10.1.9.

Ejemplo Considere la siguiente tabla de distancias entre ciudades:

*	Barcelona	Madrid	Bilbao
Barcelona	0	620	594
Madrid	620	0	394
Bilbao	594	394	0

El siguiente programa hallará la distancia entre dos ciudades de la tabla:

```
100 DATA "*","ROCKVILLE","HARTLAND","SOUTHVILLE"
110 DATA "ROCKVILLE","0","679","458"
120 DATA "HARTLAND","679","0","800"
130 DATA "SOUTHVILLE","458","800","0"
140 DIM R$(4,4)
150 FOR I=1 TO 4
160 FOR J=1 TO 4
170 READ R$(I,J)
175 NEXT J
178 NEXT I
180 INPUT "QUE PUEBLO ABANDONA? ";X$
190 INPUT "A QUE PUEBLO SE DIRIGE? ";Y$
195 REM ALGORITMO VIII
198 LET L1=0
200 FOR I=1 TO 4
210 IF X$=R$(I,1) THEN LET L1=I :LET I=4
230 NEXT I
240 IF L1=0 THEN GOTO 300
245 REM ALGORITMO DEL EJERCICIO 10.1.9
250 LET L2=0
260 FOR I=1 TO 4
270 IF Y$=R$(1,I) THEN LET L2=I :LET I=4
290 NEXT I
300 IF L2=0 THEN PRINT "LO SIENTO, ESA CIUDAD NO
    EXISTE ":GOTO 330
310 PRINT "LA DISTANCIA ES ";R$ (L1,L2)
330 END
```

```
]RUN
DE QUE CIUDAD SALE?: BILBAO
A QUE CIUDAD VA A VIAJAR?: MADRID
LA DISTANCIA A RECORRER ES DE 394
```

ALGORITMO IX: BUSQUEDA EN CUALQUIER CAMPO DE UN REGISTRO

```
REM R$( ) ES LA MATRIZ QUE CONTIENE LOS REGISTROS
REM A ES LA POSICION DEL PRIMER CARACTER DEL REGISTRO
REM F ES LA LONGITUD DEL CAMPO BUSCADO
REM N ES EL NUMERO DE REGISTROS EN LA MATRIZ
LET L=0
FOR I=1 TO N
IF S$=MID$(R$(I),A,F) THEN LET L=I :LET I=N
NEXT I
IF L=0 THEN (instrucción para el caso de que no se halle la cadena)
```

Ejemplo

El Conservatorio de Música necesita un programa que:

1. Cree un archivo (matriz) que contenga el nombre de cada estudiante, su número de matrícula, y su número de teléfono.
2. Imprima el nombre y el número de teléfono cuando le sea introducido el número de matrícula.

Este es el programa:

```
100 INPUT "¿CUANTOS ESTUDIANTES HAY? ";N
110 DIM R$(N)
120 LET B$="                    "
130 FOR I=1 TO N
140 PRINT
150 PRINT "ESTUDIANTE ";I
160 INPUT "APELLIDO: ";A$
170 INPUT "NOMBRE: ";N$
180 INPUT "NUMERO DE MATRICULA: ";M$
190 INPUT "TELEFONO: ";T$
200 LET A$=LEFT$(A$+B$,15)
210 LET N$=LEFT$(N$+B$,10)
220 LET M$=LEFT$(M$+B$,5)
230 LET T$=LEFT$(T$+B$,8)
240 LET R$(I)=A$+N$+M$+T$
250 NEXT I
260 PRINT
270 INPUT "INTRODUZCA LA MATRICULA DESEADA: ";D$
280 LET D$=LEFT$(D$+B$,5)
290 REM COMIENZO DEL ALGORITMO IX
300 LET A=0
310 FOR I=1 TO N
320 IF D$=MID$(R$(I),26,5) THEN LET A=1 :LET I=N
330 NEXT I
```

```
340 IF A=0 THEN PRINT "NO EXISTE NADIE
    CON ESA MATRICULA" :GOTO 370
350 PRINT "EL NOMBRE Y EL TELEFONO SON:"
360 PRINT LEFT$(R$(A),15);" ";MID$(R$(A),16,10);
    TAB(30);MID$(R$(A),31,8)
370 END
```

```
]RUN
CUANTOS ESTUDIANTES HAY?: 2

ESTUDIANTE 1
APELLIDO: RODRIGUEZ
NOMBRE: BLAS
NUMERO DE MATRICULA: 12345
TELEFONO: 555-3208

ESTUDIANTE 2
APELLIDO: OLMEDO
NOMBRE: JAVIER
NUMERO DE MATRICULA: 32552
TELEFONO: 555-2255

INTRODUZCA LA MATRICULA DESEADA: 12345
EL NOMBRE Y EL TELEFONO SON:
RODRIGUEZ       BLAS        555-3208
```

10.1

Ejercicios (cont.)

10. Modifique el programa anterior de tal manera que se imprima el registro del estudiante buscándolo por el número de teléfono.

10.2

Módulos y programas

Escribir un programa es como escribir un libro o un ensayo. Lo primero a realizar es un esquema de lo que el programa debe hacer. Después, ordenar los ítems en el esquema, de manera que resulte una progresión lógica. Si ha analizado bien el problema no debe ser difícil hacer unas divisiones naturales en el programa que agrupen las tareas que están relacionadas. A cada grupo le llamaremos *módulo*. Los módulos son grupos de uno o más algoritmos.

Un buen programa es consecuencia de un buen *esquema modular*, que es una lista ordenada de todos los módulos que forman el programa. En los ejemplos que siguen a continuación mostraremos primero el esquema modular, y a continuación los módulos escritos en Applesoft BASIC.

Ejemplo El Gran Hospital necesita un programa que lleve el registro de los pacientes. Las tareas a realizar se pueden dividir en cuatro módulos:

ESQUEMA MODULAR

I. ENTRADA DE DATOS DE LOS PACIENTES INGRESADOS.
II. ORDENACION DE LA MATRIZ SEGUN LOS APELLIDOS.
III. LISTADO DE LOS NOMBRES DE LOS PACIENTES.
IV. BUSQUEDA DE LOS DATOS DE LOS PACIENTES POR LOS APELLIDOS.

El programa es el que sigue (obsérvese que tres de los algoritmos se pueden encontrar en la sección 10.1):

MODULO I

```
100 REM MODULO I (ALGORITMO V)
110 INPUT "NUMERO DE PACIENTES INGRESADOS ";N
120 DIM R$(N)
130 LET B$ = "                                "
140 FOR I = 1 TO N
150 PRINT
160 PRINT "PACIENTE ";I
170 INPUT "APELLIDO: ";A$
180 INPUT "NOMBRE: ";N$
190 INPUT "DOMICILIO: ";D$
200 INPUT "TELEFONO: ";T$
210 INPUT "HABITACION ASIGNADA: ";H$
220 LET A$ = LEFT$(A$ + B$,15)
230 LET N$ = LEFT$(N$ + B$,10)
240 LET D$ = LEFT$(D$ + B$,30)
250 LET T$ = LEFT$(T$ + B$,7)
260 LET H$ = LEFT$(H$ + B$,3)
270 LET R$(I) = A$ + N$ + D$ + T$ + H$
280 NEXT I
290 REM
```

MODULO II

```
300 REM MODULO II (ALGORITMO VII)
310 FOR K = 1 TO N-1
320 FOR J = 1 TO N-K
330 IF LEFT$(R$(J),15) > LEFT$(R$(J + 1),15) THEN
    LET A$ = R$(J): LET R$(J) = R$(J + 1):LET R$(J + 1) = A$
340 NEXT J
350 NEXT K
360 REM
```

```
III
370 REM MODULO III
380 PRINT "ESTA ES LA LISTA DE LOS NOMBRES DE LOS
    PACIENTES"
390 FOR I=1 TO N
400 PRINT LEFT$(R$(I),15);", ";MID$(R$(I),16,10)
410 NEXT I
420 REM
IV
430 REM MODULO IV (ALGORITMO IX)
440 PRINT
450 INPUT "INTRODUZCA EL APELLIDO DEL PACIENTE
    BUSCADO: ";N$
460 LET N$=LEFT$(N$+B$,15)
470 LET A=0
480 FOR I=1 TO N
490 IF N$=LEFT$(R$(I),15) THEN LET A=1 :LET I=N
500 NEXT I
510 IF L=0 THEN PRINT "LO SIENTO; EL PACIENTE
    NO ESTA EN LA LISTA" : GOTO 530
520 PRINT R$(L)
530 END
```

Ejemplo La residencia de ancianos Shady Lane necesita un programa BASIC que calcule e imprima la edad media de los residentes, imprima la edad y los nombres de todos los residentes ordenados alfabéticamente por apellidos, e imprima un asterisco al lado del nombre de aquellos que sobrepasen la edad media.

ESQUEMA MODULAR

I. LECTURA DE DATOS.
II. CALCULO E IMPRESION DE LA EDAD MEDIA.
III. SEÑALIZACION DE LOS RESIDENTES QUE EXCEDAN LA EDAD MEDIA.
IV. ORDENACION ALFABETICA POR APELLIDOS.
V. IMPRESION DE LA LISTA ORDENADA.

A continuación escribiremos los módulos por separado:

MODULO I

```
10 REM MODULO I - LECTURA DE DATOS
20 REM N ES EL NUMERO DE RESIDENTES
30 READ N
```

```
 40 DIM T$(N)
 50 LET B$ = "                    "
 52 FOR I = 1 TO N
 57 REM LECTURA DE APELLIDO, NOMBRE Y EDAD
 60 READ A$,N$,E$
 62 LET A$ = LEFT$(A$ + B$,15)
 65 LET F$ = LEFT$ (N$ + B$,10)
 67 LET A$ = RIGHT$(B$ + E$,3)
 70 LET T$(I) = A$ + N$ + E$
 80 NEXT I
 85 REM NUMERO DE RESIDENTES EN LINEA 90
 90 DATA 53
100 REM A CONTINUACION LOS NOMBRES Y EDADES
    DE LOS RESIDENTES
105 DATA "KENT","CLARK","104","LANE","LOUISE","89"
107 DATA "OALD","CHIAN","72","VON BRENT","KLAUS","74"
110 (más DATA...)
120 REM FIN DEL MODULO I
```

EXPLICACION DEL MODULO I

Cada elemento de la matriz de cadenas T$ contiene un registro de información. Cada registro tiene tres campos fijos: el apellido (quince caracteres), el nombre (diez caracteres), y la edad (tres caracteres). Esta última ha sido ajustada por la derecha a su campo, como debe hacerse con todos los datos numéricos.

10.2

Ejercicios

1. ¿Cuál es el algoritmo de la sección 10.1 que es similar al módulo I, salvo porque los datos son introducidos desde teclado?

MODULO II

```
130 REM MODULO II - CALCULO E IMPRESION DE LA EDAD
      MEDIA
135 LET S = 0
140 FOR I = 1 TO N
160 LET S = S + VAL(RIGHT$(T$(I),3))
170 NEXT I
```

```
180 LET A=S/N
190 PRINT "LA EDAD MEDIA DE TODOS LOS RESIDENTES
    ES ";A
200 REM FIN DEL MODULO II
```

EXPLICACION DEL MODULO II

Aquí sumamos las edades de todos los residentes y el resultado lo dividimos por el número de ellos. La función VAL en la línea 160 cambia la edad de cadena a número para que pueda ser sumada con S (véase el capítulo II para mayor información sobre la función VAL).

10.2

Ejercicios (cont.)

2. Identifique el algoritmo de la sección 10.1 que se utiliza en el módulo II.

MODULO III

```
210 REM MODULO III - SEÑALIZA LOS RESIDENTES CUYA EDAD EXCEDE
220 REM DE LA MEDIA CON UN ASTERISCO
230 FOR I=1 TO N
240 IF (VAL(RIGHT$(T$(I),3))>A) THEN LET T$(I)=T$(I)+"*"
250 NEXT I
260 REM FIN DEL MODULO III
```

EXPLICACION DEL MODULO III

En sí, este módulo es un algoritmo de búsqueda. Un bucle repasa de nuevo las edades, comparándolas esta vez con la edad media A previamente calculada. Para marcar las edades que exceden de la media se concatena un asterisco al registro.

MODULO IV

```
270 REM MODULO IV - ORDENACION DE LOS DATOS SEGUN
    EL APELLIDO
280 FOR I=1 TO N-1
290 FOR J=1 TO N-1
310 IF LEFT$(T$(J),15)<=LEFT$(T$(J+1),15) THEN GOTO 350
```

```
320 LET X$ = T$(J)
330 LET T$(J) = T$(J + 1)
340 LET T$(J + 1) = X$
350 NEXT J
360 NEXT I
370 REM FIN DEL MODULO IV
```

EXPLICACION DEL MODULO IV

Esta es la versión alternativa de la ordenación ascendente (ALGORITMO VIa) que se vio en la sección 10.1. Utilizamos la versión alternativa simplemente por cuestión de variedad.

MODULO V

```
380 REM MODULO V - IMPRESION DE LA TABLA ORDENADA
390 REM IMPRESION DE UNA CABECERA
400 PRINT "APELLIDO";TAB(15);"NOMBRE";TAB(35);"EDAD"
410 FOR I = 1 TO N
420 PRINT LEFT$(T$(I),15);TAB(17);MID$(T$(I),16,10);
      TAB(35);MID$(T$(I),26)
430 NEXT I
440 REM FIN DEL MODULO V
450 REM Y FIN DEL PROGRAMA!
460 END
```

EXPLICACION DEL MODULO V

Para la impresión del campo correspondiente a la edad, hemos elegido la función MID$ en vez de la RIGHT$. Esto es debido a que algunos registros son más largos que otros a causa de la concatenación del asterisco. Se pueden recuperar todos los caracteres desde la posición 26 hasta el final de la cadena, utilizando la sentencia MID$(T$(I),26).

10.2

Ejercicios (cont.)

3. Escriba un programa para Shady Lane que realice dos listados de los residentes, uno ordenado por edades y otro por nombres de pila.

4. Escriba un programa para Shady Lane que realice un listado por edades. En el mismo listado deben aparecer ordenadas alfabéticamente, según apellidos, todas las personas que tengan la misma edad. *Recomendación:* De cada registro extraiga la edad y el apellido. Concatene estos dos campos (la edad primero). Realice una ordenación según este nuevo campo "híbrido".

5. La señora Smith tiene una biblioteca bastante amplia. Escríbale un programa con el siguiente esquema modular:

   I. Crear un registro para cada libro con los siguientes campos:

      a) Apellido del autor.
      b) Nombre del autor.
      c) Título del libro.
      d) Editorial.
      e) Precio.

   II. Imprimir dos listados, uno ordenado por apellidos y otro por títulos.

   III. Imprimir todos los registros de los libros cuyo precio no exceda de X, donde X es asignado mediante una sentencia INPUT.

Ejemplo La revista *Opera Times* acepta anuncios por palabras al precio de 5500 pesetas por línea, donde se considera una línea formada por 35 caracteres, espacios incluidos. Cualquier fracción de línea es considerada, a efectos de pago, como línea entera. Se debe hacer un programa BASIC que calcule el precio total de cualquier anuncio.

Por ejemplo, si el anuncio tiene 87 caracteres, espacios incluidos, el número de líneas debe ser calculado de la siguiente manera:

87 dividido entre 35 resulta 2, con un resto de 17

Dado que la línea parcial es cargada como completa, el número de líneas utilizado es de tres. El precio total es tres veces 5500 pesetas, es decir, 16500 pesetas.

El siguiente esquema se utilizará para escribir el programa BASIC:

ESQUEMA MODULAR

I. ENTRADA DEL ANUNCIO.
II. CALCULO DEL NUMERO DE CARACTERES, ESPACIOS INCLUIDOS.
III. CALCULO DEL NUMERO DE LINEAS Y DEL PRECIO.
IV. IMPRESION DEL ANUNCIO, NUMERO DE LINEAS Y COSTE TOTAL

El programa es el siguiente:

```
 10 REM CALCULO DE COSTES DE PUBLICIDAD EN EL OPERA
    TIMES
 15 REM
 20 REM TABLA DE VARIABLES
 30 REM X$ ES EL ANUNCIO
 40 REM L ES EL NUMERO DE CARACTERES EN EL ANUNCIO
 50 REM N ES EL NUMERO DE LINEAS COMPLETAS EN EL
    ANUNCIO
 60 REM S ES EL NUMERO DE LINEAS QUE SE CONSIDERAN
    PARA CALCULAR EL PRECIO
 70 REM C ES EL PRECIO TOTAL DEL ANUNCIO
 80 REM
100 REM MODULO I - ENTRADA DEL ANUNCIO
110 PRINT "ESCRIBA EL ANUNCIO. SIN COMAS NI PUNTOS Y
    COMAS"
120 INPUT X$
125 REM FIN DEL MODULO I
128 REM
130 REM MODULO II - CUENTA EL NUMERO DE CARACTERES Y
140 REM ESPACIOS DE ANUNCIO
150 LET L=LEN(S$)
155 REM FIN DEL MODULO II
158 REM
160 REM MODULO III - CALCULO DEL NUMERO DE LINEAS
165 REM Y DEL COSTE TOTAL
170 LET N=INT(L/35)
180 IF L=35*N THEN LET S=N
185 IF L>35*N THEN LET S=N+1
190 LET C=5500*S
195 REM FIN DEL MODULO III
198 REM
200 REM MODULO IV - ESCRIBE EL ANUNCIO, EL NUMERO
    DE LINEAS Y EL COSTE
210 PRINT
220 PRINT "EL NUMERO DE LINEAS ES ";S
230 PRINT "EL PRECIO DEL ANUNCIO ES $";C
240 PRINT "EL ANUNCIO DICE: " :PRINT
250 FOR I=1 TO S
260 PRINT MID$(X$,35*I-34,35)
270 NEXT I
280 REM FIN DEL MODULO IV
290 END
```

```
]RUN

ESCRIBA EL ANUNCIO, SIN COMAS NI PUNTOS Y COMAS!
?SE VENDE ORDENADOR PERSONAL - - 2000 O MEJOR
OFERTA. INCLUYE MONITOR E IMPRESORA.
LLAMAR 221 65 16 DESPUES 6 PM.

EL NUMERO DE LINEAS ES 4
EL PRECIO DEL ANUNCIO ES 220
EL ANUNCIO DICE:

SE VENDE ORDENADOR PERSONAL - - 2000
O MEJOR OFERTA. INCLUYE MONITOR E
IMPRESORA. LLAMAR 221 65 16 DESPUES
6 PM.
```

10.2

Ejercicios (cont.)

6. Modifique el programa de manera que si la longitud de la última línea es menor que media línea entera, el precio cargado por esta última línea sea sólo de 2800 pesetas; en caso contrario, debe cargarse el precio total de 5500 pesetas por la línea.

7. Suponga que la revista tiene una tarifa de precios por la cual si el anuncio excede de cuatro líneas, las que excedan se cobrarán a 4500 pesetas. Modifique el esquema y el programa BASIC de tal manera que calcule el precio total modificado.

Supongamos que la revista cambia su tarifa de precios de tal manera que sólo cobra los caracteres, y no los espacios en blanco del anuncio. Este cambio de precios requiere un cambio en el módulo II. Después de haber efectuado los cambios, el módulo queda así:

II. CALCULO DEL NUMERO DE CARACTERES NO BLANCOS DEL ANUNCIO

Por desgracia, la función LEN(X$) cuenta también los espacios en blanco. Por tanto, tenemos que modificar el módulo II de tal manera que no los cuente. Nos será útil la siguiente modificación:

```
130 REM MODULO II - CUENTA EL NUMERO DE CARACTERES
    NO BLANCOS
132 LET L=0
135 FOR I=1 TO LEN(X$)
140 IF MID$(X$,I,1)<>" " THEN LET L=L+1
145 NEXT I
150 REM FIN DEL MODULO II
```

¡FIJESE QUE NO SE HAN HECHO VARIACIONES EN EL PROGRAMA! Esto muestra el poder de la programación modular.

10.3

Las sentencias GOSUB y RETURN y las subrutinas

A veces es necesario en un programa usar el mismo proceso más de una vez. Fíjese que en el siguiente programa los módulos II, III, y IV son esencialmente iguales.

Ejemplo El señor Calvo, viajante de una firma de ordenadores, viaja frecuentemente entre tres ciudades. En su último viaje fue de Madrid a Barcelona, de aquí a Bilbao, y de esta ciudad a Madrid. He aquí un esquema modular que calcula la distancia total recorrida.

ESQUEMA MODULAR

I. ALMACENAR LAS DISTANCIAS ENTRE LAS CIUDADES EN UNA MATRIZ BIDIMENSIONAL.
II. HALLAR LA DISTANCIA ENTRE MADRID Y BARCELONA.
III. HALLAR LA DISTANCIA ENTRE BARCELONA Y BILBAO.
IV. HALLAR LA DISTANCIA ENTRE BILBAO Y MADRID.
V. SUMAR LAS DISTANCIAS E IMPRIMIR EL RESULTADO.

A medida que estudie el programa que fue escrito a partir del esquema modular anterior se dará cuenta de las similitudes entre los módulos II, III, y IV.

MODULO I

```
100 REM MODULO I - ALMACENAMIENTO DE DATOS
110 DATA "*","BARCELONA","MADRID","BILBAO"
120 DATA "BARCELONA","0","620","594"
130 DATA "MADRID","620","0","394"
140 DATA "BILBAO","594","394","0"
150 DIM R$(4,4)
160 FOR I=1 TO 4
170 FOR J=1 TO 4
180 READ R$(I,J)
190 NEXT J
200 NEXT I
210 REM
```

```
220 REM MODULO II - MADRID A BARCELONA
230 LET X$ = "MADRID"
240 LET Y$ = "BARCELONA"
250 FOR I = 1 TO 4
260 IF X$ = R$(I,1) THEN LET L1 = I
270 NEXT I
280 FOR I = 1 TO 4
290 IF Y$ = R$(1,I) THEN LET L2 = I
300 NEXT I
310 LET D1 = VAL(R$(L1,L2))
320 REM
330 REM MODULO III - BARCELONA A BILBAO
340 LET X$ = "BARCELONA"
350 LET Y$ = "BILBAO"
360 FOR I = 1 TO 4
370 X$ = R$(I,1) THEN LET L1 = I
380 NEXT I
390 FOR I = 1 TO 4
400 IF Y$ = R$(1,I) THEN LET L2 = I
410 NEXT I
420 LET D2 = VAL(R$(L1,L2))
430 REM
440 REM MODULO IV - BILBAO A MADRID
450 LET X$ = "BILBAO"
460 LET Y$ = "MADRID"
470 FOR I = 1 TO 4
480 IF X$ = R$(I,1) THEN LET L1 = I
490 NEXT I
500 FOR I = 1 TO 4
510 IF Y$ = R$(1,I) THEN LET L2 = 1
520 NEXT I
530 LET D3 = VAL(R$(L1,L2))
540 REM
550 PRINT "LA DISTANCIA TOTAL VIAJADA FUE ";D1 + D2 + D3
560 END
```

II

III

IV

V

El algoritmo situado entre las líneas 250 y 300 aparece de nuevo entre las líneas 360 y 410, y otra vez entre las 470 y 520. Podíamos haber escrito una vez el algoritmo y haberlo utilizado varias veces saltando de adelante hacia atrás, pero pronto el programa se hubiera convertido en un revoltijo de sentencias GOTO.

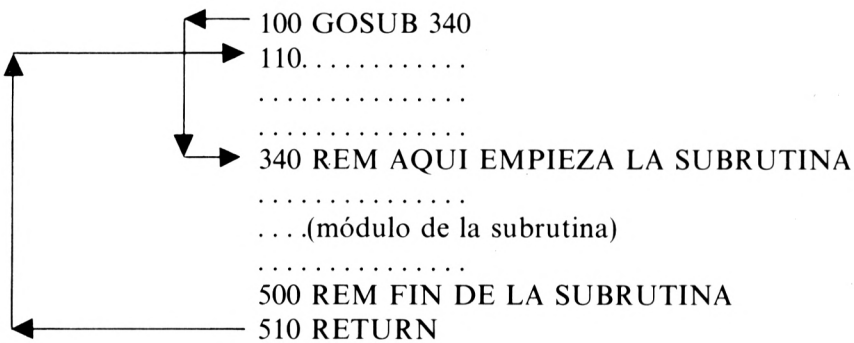
Las sentencias GOSUB y RETURN resuelven este problema.

La sentencia GOSUB es de alguna manera similar a la GOTO, ya que

las dos producen un salto incondicional a una determinada línea de programa. Por ejemplo, la sentencia

100 GOSUB 340

produce un salto a la línea 340 del programa. La gran diferencia consiste en que la instrucción GOSUB se utiliza siempre en conexión con la instrucción RETURN. Cuando se encuentra una sentencia RETURN, el ordenador salta a la sentencia GOSUB original y el proceso continúa con la siguiente instrucción al GOSUB. El programa realiza una especie de "viaje de ida y vuelta". El grupo de instrucciones (o módulo) que contiene a la sentencia RETURN se llama SUBRUTINA. Este diagrama ayudará a clarificar ideas:



Cuando se ejecuta la línea 100, el programa salta a la subrutina en la línea 340. Las flechas muestran que la sentencia RETURN devuelve el control a la línea 110, que es la siguiente a la sentencia GOSUB. (Si la línea 100 tuviera múltiples sentencias separadas por dos puntos, la sentencia RETURN hubiera devuelto el control a la sentencia de la línea 100 inmediatamente posterior a la sentencia GOSUB 340.)

Ahora escribiremos el programa del viajante de comecio usando una subrutina para calcular las distancias entre dos ciudades:

MODULO I

```
100 REM MODULO I - ALMACENAMIENTO DE DATOS
110 DATA "*","BARCELONA","MADRID","BILBAO"
120 DATA "BARCELONA","0","620","594"
130 DATA "MADRID","620","0","394"
140 DATA "BILBAO","594","394","0"
150 DIM R$(4,4)
160 FOR I=1 TO 4
170 FOR J=1 TO 4
180 READ R$(I,J)
190 NEXT J
200 NEXT I
```

```
210 REM
220 REM MODULO II - MADRID A BARCELONA
230 LET X$="MADRID"
240 LET Y$="BARCELONA"
250 GOSUB 500
260 LET D1=D
270 REM
280 REM MODULO III - BARCELONA A BILBAO
290 LET X$="BARCELONA"
300 LET Y$="BILBAO"
310 GOSUB 500
320 LET D2=D
330 REM
340 REM MODULO IV - BILBAO A MADRID
350 LET X$="BILBAO"
360 LET Y$="MADRID"
370 GOSUB 500
380 LET D3=D
390 REM
400 PRINT "LA DISTANCIA TOTAL VIAJADA FUE ";D1+D2+D3
410 GOTO 590
500 REM SUBRUTINA
510 FOR I=1 TO 4
520 IF X$=R$(1,I) THEN LET L1=I
530 NEXT I
540 FOR I=1 TO 4
550 IF Y$=R$(I,1) THEN LET L2=I
560 NEXT I
570 LET D=VAL(R$(L1,L2))
580 RETURN
585 REM
590 END
```

II

III

IV

V

SUBRUTINA

```
]RUN
LA DISTANCIA TOTAL VIAJADA FUE 1937
```

La única forma de acceder a una subrutina es haciendo un salto desde una sentencia GOSUB. La línea 410 del programa anterior es para "evitar caer" en la subrutina. Sin esta línea se produciría un error del tipo "RETURN WITHOUT GOSUB" (RETURN SIN GOSUB) al llegar a la línea 580, y el programa "reventaría" (léase: se pararía por el error).

Ejemplo Es necesario un programa para la residencia de ancianos Shady Lane que produzca dos listados: uno es una lista de los residentes ordenados

por apellidos, y el otro es una lista de los residentes ordenados por edades. Seguiremos el siguiente esquema modular:

ESQUEMA MODULAR

I. ENTRADA DE LOS NOMBRES Y EDADES DE LOS RESIDENTES EN UNA MATRIZ.
II. ORDENACION DE LA MATRIZ POR APELLIDOS.
III. LISTADO.
IV. ORDENACION DE LA MATRIZ SEGUN EDADES.
V. LISTADO.

Las tareas de ordenación y listado se escribirán como subrutinas:

```
100 REM MODULO I - ASIGNACION DE NOMBRES Y EDADES
110 INPUT "CUANTOS RESIDENTES? ";N
120 DIM R$(N)
130 LET B$ = "                    "
140 FOR I = 1 TO N
145 PRINT
150 PRINT "RESIDENTE NUMERO    ";I
160 INPUT "APELLIDO: ";A$
170 INPUT "NOMBRE: ";N$
180 INPUT "EDAD: ";E$
190 LET R$(I) = LEFT$(A$ + B$,15) + LEFT$(N$ + B$,10) +
    RIGHT$(B$ + E$,3)
200 NEXT I
210 REM
220 REM MODULO II - ORDENACION POR APELLIDOS
230 LET K = 1
240 LET Q = 15
250 GOSUB 1000
255 REM
260 MODULO III - LISTADO
265 PRINT : PRINT "ORDENACION POR APELLIDOS"
270 GOSUB 2000
280 REM
290 REM MODULO IV - ORDENACION POR EDADES
300 LET K = 26
310 LET Q = 3
320 GOSUB 1000
340 REM MODULO V - LISTADO
```

```
345 PRINT :PRINT "ORDENACION POR EDADES"
350 GOSUB 1000
360 REM
370 GOTO 9999
380 REM
1000 REM SUBRUTINA DE ORDENACION SEGUN CUALQUIER
     CAMPO
1010 FOR I=1 TO N-1
1020 FOR J=1 TO N-1
1030 IF MID$(R$(J),K,Q)>MID$(R$(J+1),K,Q) THEN
     LET A$=R$(J) :LET R$(J)=R$(J+1) :LET R$(J+1)=A$
1040 NEXT J
1050 NEXT I
1060 RETURN
1070 REM
2000 REM SUBRUTINA DE LISTADO
2010 PRINT
2015 PRINT "APELLIDO"; TAB(16); "NOMBRE";
     TAB (30); "EDAD"
2020 FOR I=1 TO N
2030 PRINT LEFT$(R$(I),15); TAB(16); MID$(R$(I),16,10);
     TAB(30); RIGHT$(R$(I),3)
2040 NEXT I
2050 RETURN
2060 REM
9999 END
```

```
]RUN
CUANTOS RESIDENTES? 3

RESIDENTE NUMERO 1
APELLIDO: SAENZ
NOMBRE: ANA
EDAD: 92

RESIDENTE NUMERO 2
APELLIDO: DURCAL
NOMBRE: L.
EDAD: 72

RESIDENTE NUMERO 3
APELLIDO: OLMO
NOMBRE: RAMON
EDAD: 104
```

```
ORDENACION POR NOMBRES

APELLIDO        NOMBRE          EDAD
DURCAL          L.                72
OLMO            RAMON            104
SAENZ           ANA               92

ORDENACION POR EDADES

APELLIDO        NOMBRE          EDAD
DURCAL          L.                72
SAENZ           ANA               92
OLMO            RAMON            104
```

10.3

Ejercicio

1. Modifique el programa para que la ordenación se haga según el nombre completo, apellido y nombre combinados.

10.4

ON-GOTO, ON-GOSUB, y el menú

Si la sentencia IF-THEN se combina con una sentencia de ramificación como GOTO o GOSUB, creamos una sentencia que se llama ramificación o salto condicional. Este también se puede realizar con las sentencias ON-GOTO y ON-GOSUB.

La siguiente secuencia de IF-THEN GOTO envía el control a varios puntos del programa, dependiendo del valor de K:

```
100 IF K=1 THEN GOTO 250
110 IF K=2 THEN GOTO 500
120 IF K=3 THEN GOTO 355
130 IF K=4 THEN GOTO 999
```

El propósito de la sentencia ON-GOTO es condensar las anteriores sentencias en una, como la siguiente:

```
100 ON K GOTO 250, 500, 355, 999
```

El valor de K determina dónde debe ir el control del programa. Si K es igual a 1, el control pasa a la línea 250, que es la primera de las listadas; si K es igual a 2 el control pasa a la línea 500, que es la segunda, y así sucesivamente. Digamos que K actúa como un índice de los números de línea de la lista. Si K es menor que 1 o mayor que la cantidad de números de línea, entonces el control pasa a la siguiente sentencia a la ON-GOTO.

La sentencia ON-GOSUB se comporta de igual manera, excepto que el control se transfiere a una subrutina. El RETURN de la subrutina devolverá el control a la siguiente línea a la sentencia ON-GOSUB.

El verdadero valor de las sentencias ON-GOSUB reside en su utilización en "menús". Decimos que un programa está organizado por menú si las distintas funciones que realiza el programa pueden ser elegidas por el usuario de una lista de opciones. A esta lista se le llama menú. El siguiente ejemplo muestra un programa organizado por menú.

Ejemplo El colegio Green Bay necesita un programa como el descrito a continuación:

I. ALMACENAMIENTO EN UNA MATRIZ DEL NUMERO DE ALUMNO, APELLIDO, NOMBRE Y EDAD DE CADA ALUMNO.
II. BUSQUEDA DE LA FICHA POR EL NUMERO DE ALUMNO (MATRICULA).
III. BUSQUEDA DE LA FICHA POR EL APELLIDO.

MENU

```
10 REM LAS LINEAS 20 A 80 FORMAN EL MENU
20 PRINT "1. ALMACENAR LA INFORMACION
   DEL ESTUDIANTE"
30 PRINT "2. BUSQUEDA POR NUMERO DE MATRICULA"
40 PRINT "3. BUSQUEDA POR APELLIDOS"
50 PRINT "4. ACABAR"
60 INPUT "INTRODUZCA EL NUMERO DE LA OPCION
   DESEADA: ";K
70 ON K GOSUB 90,190,260,330
80 GOTO 10
85 REM
```

MODULO I

```
90 REM MODULO I - ALMACENAMIENTO DE DATOS
100 INPUT "CUANTOS ALUMNOS HAY
    MATRICULADOS? ";N
110 DIM S$(N,4)
120 FOR I=1 TO N
130 INPUT "NUMERO DE MATRICULA ";S$(I,1)
140 INPUT "APELLIDO ";S$(I,2)
150 INPUT "NOMBRE ";S$(I,3)
160 INPUT "EDAD ";S$(I,4)
170 NEXT I
180 RETURN
185 REM
```

```
II
190 REM MODULO II - BUSQUEDA POR NUMERO
    DE MATRICULA
200 INPUT "INTRODUZCA UN NUMERO DE MATRICULA: ";Q$
205 LET L=0
210 FOR I=1 TO N
220 IF Q$=S$(I,1) THEN LET L=I
230 NEXT I
235 IF L<>0 THEN PRINT S$(L,1), S$(L,2), S$(L,3),
    S$(L,4)
240 RETURN
250 REM
III
260 REM MODULO III - BUSQUEDA POR APELLIDOS
270 INPUT "INTRODUZCA UN APELLIDO: ";Q$
275 LET L=0
280 FOR I=1 TO N
290 IF Q$=S$(I,2) THEN L=I
300 NEXT I
310 IF L<>0 THEN PRINT S$(L,1), S$(L,2), S$(L,3),
    S$(L,4)
320 RETURN
330 END
```

Fíjese que cada RETURN devuelve el control a la línea 80, la cual produce un salto a la línea 10, al principio del menú.

10.4

Ejercicios

1. Fíjese que los módulos II y III son esencialmente iguales, y pueden ser sustituidos por una subrutina. Revise el programa anterior y haga una subrutina que sustituya a los dos módulos.

2. Este programa tiene serios defectos, entre ellos que si varios alumnos tienen el mismo apellido, sólo aparecerán los datos de uno de ellos. Modifique el programa de tal manera que si hay varios estudiantes con el mismo apellido, sean escritas las fichas de todos.

10.5

Programación deductiva ("top-down"*)

La programación modular es una forma simple, pero potente, de crear una estructura en los programas BASIC. Pero incluso un programa formado por módulos puede resultar lioso si no se pone un especial cuidado en

* Top-Down significa de arriba a abajo *(N. del T.)*.

darle una forma lógica. Sugerimos una distribución del programa gracias a la cual el control fluya suavemente desde el módulo superior hacia abajo. Este es un estilo conocido como programación deductiva o TOP-DOWN. Lo que sigue es un sumario de algunas normas básicas de la programación TOP-DOWN:

1. Muévase desde el primer módulo, o índice general, siempre hacia abajo.
2. Se pueden saltar uno o más módulos en el camino descendente.
3. El flujo del programa debe ser descendente. Siempre que pueda evite moverse a un módulo superior (con excepción del índice general).
4. Cada módulo debe tener una única entrada y una única salida. El punto de entrada debe ser la primera línea del módulo, y el de salida la última. Esto hará que el flujo sea descendente.
5. El uso de la sentencia GOTO debe ser reducido al mínimo. Algunos lenguajes de programación están diseñados para no tener necesidad de usar la sentencia GOTO, pero el BASIC no es tan afortunado. Y aunque es imposible el diseño de programas sin GOTO en BASIC, le recomendamos siempre tenga presente que:

   Generalmente, las sentencias GOTO hacen al programa más difícil de leer, más difícil de entender, y más difícil de depurar. Deben ser usadas como último recurso, cuando no puedan ser sustituidas por cualquier otra sentencia BASIC.

El uso excesivo del GOTO puede crear una situación calificada como "código spaghetti", ¡pura pasta!

10.6

Documentación

Existen dos poderosas razones para documentar los programas. Primero, aunque hoy pueda estar al tanto del más minúsculo detalle de su programa, es seguro que su memoria será mucho menos portentosa de aquí a un año. Para su propio uso, y aunque su memoria sea prodigiosa, ¡documente sus programas! Segundo, otros programadores pueden estar interesados en sus programas. Suele ocurrir con alguna frecuencia, que una empresa necesite poner al día algún programa y que el autor del programa haya dejado la compañía hace años. Aunque usted piense que nunca nadie va a mirar sus programas,... ¡documéntelos!

Bien, ahora hemos dejado las cosas bastante claras. Pero aún persiste

una duda: ¿Qué es lo que debemos poner? Nunca es necesario comentar lo que es obvio. Pero, como es normal, lo que es obvio es algo bastante subjetivo. Como regla general, nosotros encontramos muy útiles en una documentación dos cosas. Una es una tabla de variables que liste todas las variables de su programa y dé una breve descripción de su utilidad. Segundo, sugerimos que sean utilizados REM (comentarios) para indicar el comienzo de cada módulo y la tarea que realiza.

Ejemplo de documentación

```
100 REM PROGRAMA PARA EL CALCULO DE BENEFICIOS
110 REM C=COMISION A LAS VENTAS
120 REM S=BRUTO DE LAS VENTAS
130 REM P=BENEFICIO NETO
140 REM MODULO I - ASIGNACION DE DATOS
    .........................
    .........................
200 REM MODULO II - CALCULO DEL BRUTO
    DE LAS VENTAS
    .........................
    .........................
500 REM MODULO III - COMISION A LAS VENTAS
    .........................
    .........................
800 REM MODULO IV - CALCULO DE LOS BENEFICIOS
    .........................
    .........................
1000 REM MODULO V - BENEFICIO NETO
    .........................
    .........................
1300 END
```

10.7

Proyectos

1. La señora Pérez, profesora de Biología, necesita representar las notas de sus alumnos. Las calificaciones están basadas en el siguiente sistema de puntuación:

85 - 100	A
70 - 84	B
55 - 69	C
40 - 54	D
0 - 39	F

Para calcular las notas finales utiliza el siguiente sistema:

1. Suma las notas individuales de cada ejercicio.
2. Añade a esta suma el resultado del examen final multiplicado por dos.
3. Divide la suma total entre el número de exámenes (incluido el final) más uno.

Escriba un programa organizado por menú que:

1. Almacene las notas obtenidas en los ejercicios y en el examen final por cada alumno.
2. Calcule las notas definitivas y los niveles.
3. Haga una ordenación alfabética por apellidos.
4. Haga una ordenación según las letras de los niveles obtenidos.
5. Imprima una lista de los estudiantes que han obtenido un determinado nivel (letra).
6. Haga un listado de los alumnos cuyo nivel es superior o inferior a uno determinado.

2. La compañía telefónica necesita un programa que realice las siguientes tareas:

1. Almacene el nombre, domicilio y número de teléfono de cada abonado.
2. Ordene los registros según:

   a) Número de teléfono.
   b) Apellidos.
   c) Distrito.

   y los imprima.
3. Dada cualquier porción de un apellido seleccionar todos los abonados cuyos apellidos empiecen por la porción fijada, y escribir sus fichas.

3. Escriba un programa para la compañía inmobiliaria Golden House que realice lo siguiente:

1. Almacene la información de la localización, el número de dormitorios, la superficie, y el precio de las casas en venta.
2. Ordene según cualquier campo elegido.
3. Imprima la tabla ya ordenada.

PRACTICA 10.1

GOSUB/RETURN

Escriba la orden: NEW

Las sentencias GOSUB y RETURN son usadas para formar una subrutina.

Escriba el programa:

```
10 GOSUB 40
20 PRINT "LINEA 20"
30 GOTO 60
40 PRINT "LINEA 40"
50 RETURN
60 END
```

Las líneas 40 y 50 forman una subrutina. El número de la primera línea de la subrutina es el número de línea dado en la sentencia GOSUB. El número de la última línea de la subrutina es el que aparece en la sentencia RETURN.

Escriba la instrucción: RUN

¿Qué respuesta da el programa? _____

_____

La sentencia 10 GOSUB 40 produce una ramificación a la línea 40. La sentencia 50 RETURN origina un salto a la línea 20.

Liste los números de línea en el orden en que son ejecutadas. _____

_____

El orden en que las sentencias son ejecutadas es: 10, 40, 50, 20, 30 y 60. La línea 10 es ejecutada primero. Entonces el programa salta a la línea 40. La ejecución de la línea 50 devuelve el control a la línea 20.

La sentencia RETURN produce un salto a la sentencia siguiente a la última GOSUB ejecutada.

Cuando se ejecuta una sentencia GOSUB, el BASIC almacena en una lista especial (llamada en inglés *stack*, pila) el número de línea de la sentencia GOSUB. Una sentencia RETURN produce un salto al último número de línea que ha sido escrito en la lista, y acto seguido se borra el número. Cada RETURN ejecutado borra un número de la lista. Por tanto, es necesario haber ejecutado una sentencia GOSUB antes de ejecutar una RETURN.

Escriba la sentencia: 30 PRINT "NUMERO DE LINEA 30"

Escriba las órdenes: LIST
RUN

¿Por qué da error el ordenador? _____

_____

_____

_____

Cuando el ordenador ejecuta la sentencia 50 RETURN por segunda vez, la pila está vacía. Para evitar caer accidentalmente en una subrutina y, por tanto, producir un error, separe sus subrutinas del programa principal. Haga esto acabando el programa con un salto al final (como en 30 GOTO 60), o simplemente utilice una sentencia END (como 30 END).

Explique por qué son necesarias las subrutinas en un programa _____

_____

_____

_____

_____

Algunas veces un grupo determinado de sentencias debe ser ejecutado varias veces en el mismo programa. Las subrutinas le permiten organizar el programa de manera que el grupo de instrucciones sólo sea escrito una vez.

PRACTICA 10.2

Las sentencias ON-GOTO y ON-GOSUB

Escriba la orden: NEW

Las sentencias ON-GOTO y ON-GOSUB son instrucciones de transferencia condicional multirramificadas.

El argumento que sigue al ON debe ser una expresión aritmética o una variable numérica. Los números de línea escritos a continuación del GOTO o del GOSUB deben estar separados por comas. Por ejemplo:

```
ON X GOTO 110,230,80
ON K+7 GOSUB 430,220,550
```

Escriba el programa:

```
10 INPUT "PULSE UN NUMERO: ";K
20 ON K GOTO 40,60,80
```

```
30 END
40 PRINT "LINEA 40"
50 GOTO 10
60 PRINT "LINEA 60"
70 GOTO 10
80 PRINT "LINEA 80"
90 GOTO 10
```

Escriba la orden: RUN

Pulse el número: 1

¿Cuál es la respuesta del programa? _____

Después de que le ha sido asignado a K el valor 1, la línea 20 transfiere el control a la línea 40.

Pulse el número: 2

¿Cuál es la respuesta del programa? _____

Después de que a K se le ha asignado el valor 2, el control es transferido a la línea 60.

Pulse el número: 3

¿Cuál es la respuesta del programa? _____

¿A qué línea ha saltado el programa después de haber sido ejecutada la línea 20?

_____

La respuesta es la línea 80.

¿Qué relación hay entre el valor asignado a K y los números de línea listados después del GOTO? _____

_____

_____

_____

Si a K se le asigna el valor 1, el control se transfiere al primer número listado; si K es 2, el control se transfiere al segundo y así sucesivamente. La variable K es un índice de la lista de números de línea.

Pulse el número: 1.7

¿Cuál es la respuesta del programa? _____

Fíjese que la salida es LINEA 40. Esto es porque el valor de K es truncado a un número entero, en este caso el 1.

¿Cuál sería la respuesta si el número pulsado fuera 3.5? _____

Pulse el número: 3.5

¿Cuál es la respuesta del programa? _____

Explique por qué la respuesta ha sido: LINEA 80. _____

_____

_____

El número 3.5 es truncado al entero 3. Por tanto, el programa transfiere el control a la tercera línea de la lista que hay después del GOTO.

Siempre que el argumento que sigue al ON sea igual a cero, o mayor que el número de líneas listadas después del GOTO o el GOSUB, el programa transfiere el control a la sentencia situada a continuación de las sentencias ON-GOTO u ON-GOSUB.

Pulse el número: 6

Explique por qué el programa se acaba. _____

_____

_____

El programa se acaba en la línea 30. Y como sólo hay tres líneas listadas detrás del GOTO, y a K se le ha asignado el valor 6, entonces el control pasa a la línea siguiente.

Escriba la orden: RUN

Escriba el número: −2

¿Cuál es la respuesta del ordenador? _____

El índice variable que sigue a ON debe ser positivo.

Escriba la instrucción: NEW

Escriba el programa:
```
10 HOME
20 PRINT "0. SUMA DE 1 A 100"
```

```
 30 PRINT "1. SUMA DE LOS NUMEROS PARES
      DE 2 A 100"
 40 PRINT "2. SUMA DE LOS NUMEROS IMPARES
      DEL 1 AL 99"
 50 PRINT "3. ACABAR"
 60 PRINT
 70 INPUT "PULSE UN NUMERO DEL MENU: ";X
 80 ON X+1 GOSUB 110,140,170,200
 90 INPUT "PULSE RETURN PARA CONTINUAR: ";X$
100 GOTO 10
110 REM SUBRUTINA SUMA DE 1 A 100
120 PRINT "LA SUMA DE LOS NUMEROS DE
      1 A 100 ES ";5050
130 RETURN
140 REM SUBRUTINA DE SUMA DE NUMEROS PARES
150 PRINT "LA SUMA DE NUMEROS PARES DE
      2 A 100 ES ";2550
160 RETURN
170 REM SUBRUTINA DE SUMA DE NUMEROS
      IMPARES
180 PRINT "LA SUMA DE NUMEROS IMPARES DE
      1 A 99 ES ";2500
190 RETURN
200 END
```

Escriba la orden: RUN

Elija un número de los del menú y escríbalo. Explique por qué es importante la línea 90.

_____

_____

La mejor forma de explicar la importancia de la línea 90 es quitarla del programa, y volver a ejecutarlo.

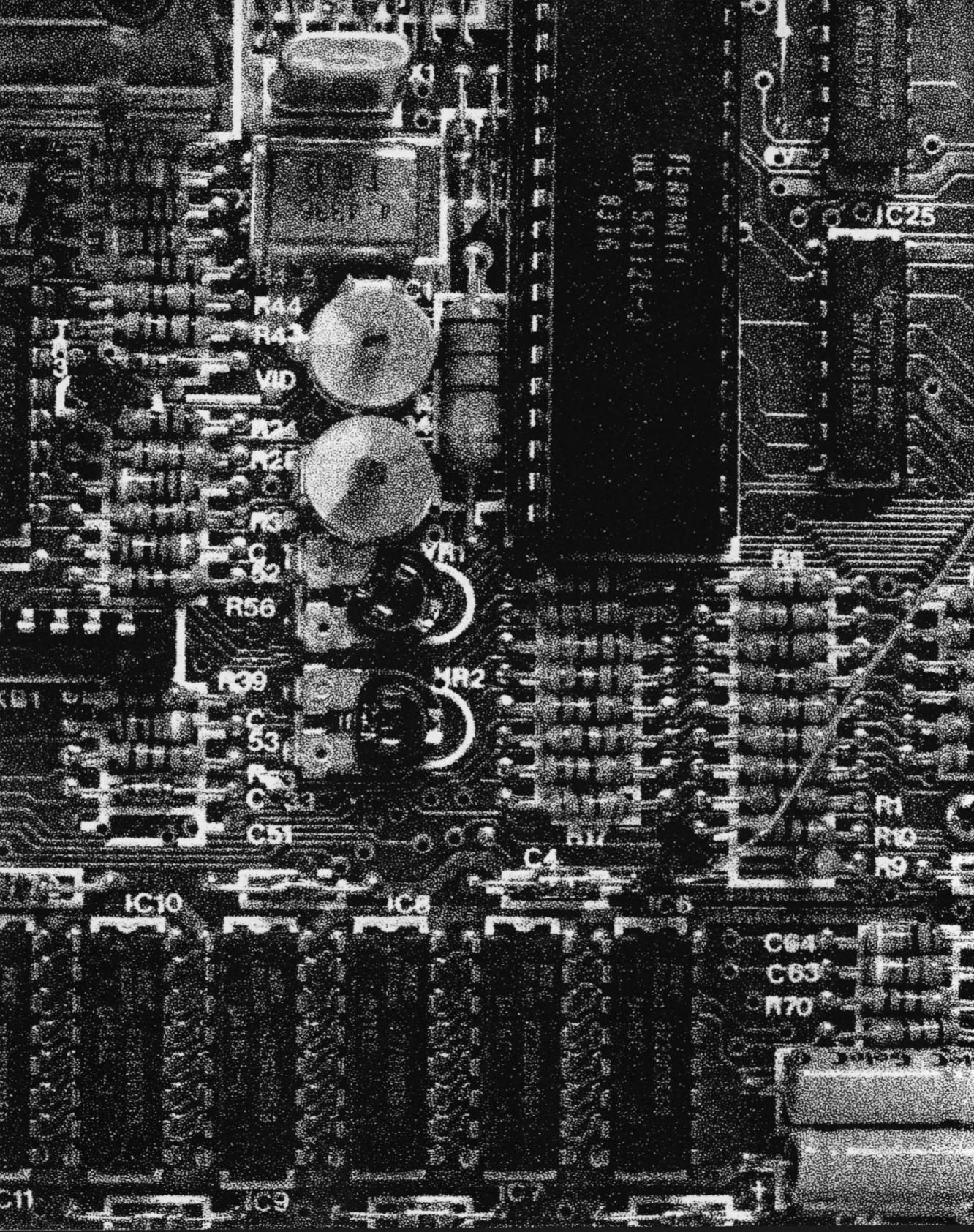
Explique por qué en la línea 80 el argumento que sigue al ON es X + 1.

_____

_____

Ya que el programa indica al usuario que elija un número entre 0 y 3, la expresión X + 1 indexará correctamente los cuatro números de línea que hay escritos detrás del GOSUB.

Hay que tener cuidado siempre que una expresión aritmética siga al ON. La expresión aritmética debe generar un número no negativo, porque si no es así, se producirá un error.


IC25
R44
VID
R56
R39
C51
C4
R1
R9
IC10
IC8
C64
C63
R70
IC9
IC7

11
Procesamiento de archivos secuenciales

11.0
Introducción

La memoria principal del ordenador es volátil, esto es, los datos se pierden cuando se apaga el ordenador. Hay muchas aplicaciones en las que los datos deben ser almacenados permanentemente para ser utilizados con posterioridad, o por otro programa. Para el almacenamiento permanente de datos se utilizan dispositivos de memoria no volátil, tales como discos o cintas magnéticas. Este capítulo es una introducción al almacenamiento de datos en *diskette*. Antes de seguir adelante, sería conveniente repasar la sección 9.3 que trata sobre campos y registros.

11.1
Escritura de registros en un archivo secuencial

REGLAS PARA LA ESCRITURA DE REGISTROS
EN UN ARCHIVO SECUENCIAL

1. Abrir el archivo.

2. Avisar al ordenador de que a continuación se va a escribir en un archivo.
3. Escribir los registros en el archivo.
4. Cerrar el archivo.

ALGORITMO X - ESCRITURA DE REGISTROS EN UN ARCHIVO DE UN DISCO:

```
REM T$(I) SON LOS REGISTROS QUE DEBEN SER ENVIADOS
   AL DISCO
PRINT CHR$(4) "OPEN nombre del archivo"
PRINT CHR$(4); "WRITE nombre del archivo"
FOR I=1 TO N
PRINT T$(I)
NEXT I
PRINT CHR$(4); "CLOSE nombre del archivo"
```

EXPLICACION DEL ALGORITMO X

PRINT CHR$(4); "OPEN nombre del archivo"

El CHR$(4) es un código especial que cuando se usa en una sentencia PRINT dirige la instrucción al DOS (Disk Operating System, Sistema Operativo de Discos). OPEN, nombre del archivo, es una orden del DOS que hace abrir el archivo con el nombre indicado (Regla 1). Si no existe este nombre, se crea un archivo nuevo (la función CHR$(X) es tratada más adelante, en el capítulo 12).

PRINT CHR$(4); "WRITE nombre del archivo"

Esta orden indica al ordenador que las siguientes instrucciones PRINT están dirigidas al disco y no a la pantalla (Regla 2).

PRINT T$(I)

Esta sentencia escribirá cada registro en el archivo adecuado del disco (Regla 3). La escritura de estos registros se realiza secuencialmente a partir del primer registro.

PRINT CHR$(4); "CLOSE nombre del archivo"

Esta sentencia cerrará el archivo y redirigirá las sentencias PRINT de nuevo al TRC (Regla 4). *Importante:* Cuando usted haya acabado de

escribir en un archivo es necesario que lo cierre, ya que si no se hace esto los datos se podrían perder. Como el dato es escrito en un archivo, es almacenado temporalmente en un *buffer*, o memoria intermedia, en el ordenador. Al cerrar el archivo se vacía el *buffer* y se escriben los datos en el archivo.

APLICACION

Considere el problema de almacenar permanentemente los nombres y las edades de los residentes de la Shady Lane. El esquema modular que resuelve el problema es el siguiente:

ESQUEMA MODULAR

I. ENTRADA DE LOS NOMBRES Y EDADES DE LOS RESIDENTES EN LOS REGISTROS.
II. ESCRITURA DE LOS REGISTROS EN UN ARCHIVO DEL DISCO.

MODULO I

```
10 REM MODULO I - LECTURA DE LOS DATOS DE SHADY
   LANE
20 INPUT "ESCRIBA EL NUMERO DE RESIDENTES: ";N
30 DIM T$(N)
35 LET B$ = "                    " :REM B$ TIENE 15 BLANCOS
40 FOR I=1 TO N
50 REM ENTRADA DE APELLIDO, NOMBRE, Y EDAD
60 INPUT "APELLIDO: ";A$
70 INPUT "NOMBRE: ";N$
80 INPUT "EDAD: ";E$
90 LET L$ = LEFT$(A$ + B$,15)
100 LET F$ = LEFT$(N$ + B$,10)
110 LET A$ = RIGHT$(B$ + E$,3)
115 LET T$(I) = A$ + N$ + E$
120 NEXT I
130 REM FIN DEL MODULO
```

El módulo II que sigue a continuación almacena todos los registros de la matriz en el disco dentro de un archivo llamado SHADY LANE, utilizando el algoritmo X.

```
135 REM MODULO II - ALMACENA LOS REGISTROS T$(I) DE
140 REM SHADY LANE EN EL DISCO EN ORDEN SECUENCIAL
160 PRINT CHR$(4); "OPEN SHADY LANE"
170 PRINT CHR$(4); "WRITE SHADY LANE"
180 FOR I=1 TO N
190 PRINT T$(I)
200 NEXT I
210 PRINT CHR$(4); "CLOSE SHADY LANE"
220 END
```

11.1

Ejercicios

1. El señor Solís ha comprado un nuevo Apple. Escriba un programa BASIC que almacene su listín telefónico personal en un disco donde cada registro debe contener los siguientes campos:

   Campo 1: Apellido
   Campo 2: Nombre
   Campo 3: Domicilio
   Campo 4: Ciudad
   Campo 5: Distrito
   Campo 6: Número de teléfono

2. La compañía de alquiler de coches RAL alquila diversos tipos de coches y camiones. Recientemente ha comprado un Apple. Escriba un programa que sirva para almacenar en un disco un archivo que contenga registros con los siguientes campos:

   Campo 1: Tipo de vehículo
   Campo 2: Fabricante
   Campo 3: Modelo
   Campo 4: Año
   Campo 5: Kilometraje recorrido
   Campo 6: Precio por día de alquiler
   Campo 7: Precio por kilómetro del alquiler

3. El señor Ginés, profesor de Informática, necesita un programa BASIC para almacenar las calificaciones de sus alumnos. Escriba un programa que almacene en un disco un archivo que contenga registros con los siguientes campos:

   Campo 1: Apellido
   Campo 2: Nombre
   Campo 3: Número de matrícula

Campo 4: Nota primer parcial
Campo 5: Nota segundo parcial
Campo 6: Nota tercer parcial
Campo 7: Nota examen final

11.2

Lectura de registros en un archivo secuencial

REGLAS PARA LA LECTURA DE REGISTROS
EN UN ARCHIVO SECUENCIAL

1. Abrir el archivo.
2. Avisar al ordenador de que a continuación se va a leer de un archivo.
3. Leer los registros de un archivo.
4. Cerrar el archivo.

El siguiente algoritmo transforma estas reglas en BASIC.

ALGORITMO XI - LECTURA DE REGISTROS DE UN ARCHIVO

```
        REM T$(I) CONTIENE LOS REGISTROS DE UN ARCHIVO
        ONERR GOTO línea B
        PRINT CHR$(4); "OPEN nombre del archivo"
        PRINT CHR$(4); "READ nombre del archivo"
        LET I=0
línea A LET I=I+1
        INPUT T$(I)
        GOTO línea A
línea B PRINT CHR$(4); "CLOSE nombre del archivo"
        POKE 216,0
```

EXPLICACION DEL ALGORITMO XI

ONERR GOTO línea B

Esta instrucción es una sentencia Applesoft, que causa un salto a la línea especificada siempre que ocurre un error. Cuando todos los registros del archivo de datos han sido leídos, y se intenta leer un nuevo

registro, se produce un error del tipo END OF DATA (fin de los datos). Este error produce un salto a la línea B por causa de la sentencia ONERR.

PRINT CHR$(4); "READ nombre del archivo"

Esta sentencia redirecciona las entradas subsiguientes para aceptar los datos del archivo dado (Regla 2).

INPUT T$(I)

Esta sentencia INPUT toma un registro del archivo en el disco, empezando por el primero, y se lo asigna a la variable T$(I).

POKE 216,0

Esta instrucción cancela la sentencia ONERR, de tal manera que errores subsiguientes no causen un salto a la línea B. (Véase el capítulo 12 para más información sobre la instrucción POKE.)

APLICACION

En un programa previo se ha creado un archivo para la residencia Shady Lane. El siguiente esquema modular leerá este archivo del disco, lo introducirá en la memoria principal del ordenador y escribirá dicho archivo en pantalla.

ESQUEMA MODULAR

I. LEER EL ARCHIVO SHADY LANE DEL DISCO.
II. ESCRIBIR EN PANTALLA EL ARCHIVO.

MODULO I

```
100 REM MODULO I - LEER EL ARCHIVO SHADY LANE DEL
    DISCO
105 DIM T$(100)
110 ONERR GOTO 170
120 PRINT CHR$(4): "OPEN SHADY LANE"
130 PRINT CHR$(4); "READ SHADY LANE"
135 LET I=0
```

```
140 LET I=I+1
150 INPUT T$(I)
160 GOTO 140
170 PRINT CHR$(4); "CLOSE SHADY LANE"
180 POKE 216,0
182 LET I=I-1 :REM I ES EL NUMERO DE REGISTROS
185 REM FIN DEL MODULO
```

El módulo II que viene a continuación escribirá en pantalla los registros T$(I).

MODULO II

```
200 REM MODULO II - ESCRITURA DE LOS REGISTROS
210 FOR K=1 TO I
220 PRINT T$(K)
230 NEXT K
240 END
```

11.2

Ejercicios

1. Añada un módulo entre el I y el II, de tal manera que los registros puedan ser ordenados por apellidos antes de ser escritos en pantalla.

2. Escriba programas para leer datos de los archivos creados en el ejercicio 11.1.

11.3

Almacenamiento de datos por campos

El ordenador Apple permite que puedan almacenarse directamente en un disco campos de un registro, sin necesidad de concatenar los campos para formar un registro. A pesar de que este método no exige que los campos tengan longitudes fijas, nosotros seguiremos manteniendo los tamaños de los campos, ya que esto es importante si se tienen que realizar ordenaciones o búsquedas en los registros. El siguiente programa es el de Shady Lane, variado de tal manera que los campos se almacenen por separado.

MODULO I

```
 10 REM MODULO I - ENTRADA DE DATOS
 20 INPUT "CUANTOS RESIDENTES HAY?: ";N
 30 DIM T$(N,3)
 35 LET B$ ="                    " :REM B$ TIENE 15 BLANCOS
 40 FOR I=1 TO N
 50 REM ENTRADA DE APELLIDO, NOMBRE, EDAD
 60 INPUT "APELLIDO: ";A$
 70 INPUT "NOMBRE: ";N$
 80 INPUT "EDAD: ";E$
 85 REM FORMATEADO DE LOS CAMPOS AL TAMAÑO
    CORRECTO
 90 LET T$(I,1)=LEFT$(A$+B$,15)
100 LET T$(I,2)=LEFT$(N$+B$,10)
110 LET T$(I,3)=RIGHT$(B$+E$,3)
120 NEXT I
130 REM FIN DEL MODULO I
```

MODULO II

```
140 REM MODULO II - ALMACENAMIENTO EN DISCO
    EN ORDEN SECUENCIAL
160 PRINT CHR$(4);"OPEN SHADY LANE"
170 PRINT CHR$(4);"WRITE SHADY LANE"
180 FOR I=1 TO N
190 PRINT T$(I,1)
200 PRINT T$(I,2)
210 PRINT T$(I,3)
215 NEXT I
220 PRINT CHR$(4);"CLOSE SHADY LANE"
230 END
```

El siguiente programa leerá los registros del archivo SHADY LANE. Se leerán tres campos de una vez y se escribirán los datos en la pantalla:

```
100 REM PROGRAMA PARA LEER REGISTROS DE SHADY LANE
125 DIM Q$(100)
130 ONERR GOTO 195
140 PRINT CHR$(4);"OPEN SHADY LANE"
150 PRINT CHR$(4);"READ SHADY LANE"
155 LET I=0
160 LET I=I+1
170 INPUT A$,N$,E$
180 LET Q$(I)=A$+N$+E$
190 GOTO 160
195 PRINT CHR$(4);"CLOSE SHADY LANE"
200 POKE 216,0
203 LET I=I-1
```

```
205 FOR K=1 TO I
210 PRINT Q$(K)
220 NEXT K
230 END
```

11.3

Ejercicios

1. Modifique los programas escritos en el ejercicio 11.1 utilizando el método de almacenamiento por campos separados.

11.4

Incorporación de registros a un archivo

Después de abrirse un archivo con la sentencia OPEN, la lectura o la escritura en el disco *siempre* empiezan por el primer registro del archivo. Esta sección trata de la tarea de añadir un registro al final de un archivo en un disco.

REGLAS PARA AÑADIR REGISTROS A UN ARCHIVO SECUENCIAL:

1. Abrir el archivo en modo actualizar ("APPEND").
2. Avisar al ordenador de que a continuación se va a escribir en un archivo.
3. Escribir el registro en el archivo.
4. Cerrar el archivo.

ALGORITMO XII - AÑADIENDO REGISTROS A UN ARCHIVO

```
PRINT CHR$(4); "APPEND nombre del archivo"
PRINT CHR$(4); "WRITE nombre del archivo"
FOR I=1 TO N
PRINT T$(I)
NEXT I
PRINT CHR$(4); "CLOSE nombre del archivo"
```

EXPLICACION DEL ALGORITMO XII

```
PRINT CHR$(4); "APPEND nombre del archivo"
```

Esta instrucción sustituye a la instrucción OPEN. El archivo se abre, pero el añadido de registros se realizará al final del archivo.

PRINT T$(I)

Esta instrucción escribirá el registro T$(I) al final del archivo.

APLICACION

Este programa añadirá registros adicionales al archivo SHADY LANE:

```
100 REM MODULO I - AÑADIDO DE UNO O MAS RESIDENTES
110 INPUT "¿CUANTOS RESIDENTES VA A AÑADIR? ";N
120 DIM T$(N)
130 LET B$ ="                    " :REM B$ TIENE 15 BLANCOS
140 INPUT "APELLIDO: ";A$
150 INPUT "NOMBRE: ";N$
160 INPUT "EDAD: ";E$
170 LET L$ = LEFT$(A$ + B$,15)
180 LET F$ = LEFT$(N$ + B$,10)
190 LET A$ = RIGHT$(B$ + E$,3)
200 LET T$(I) = A$ + N$ + E$
210 NEXT I
220 MODULO II - AÑADE N REGISTROS AL FINAL DEL ARCHIVO
230 PRINT CHR$(4); "APPEND SHADY LANE"
235 PRINT CHR$(4); "WRITE SHADY LANE"
240 FOR I = 1 TO N
250 PRINT T$(I)
260 NEXT I
270 PRINT CHR$(4); "CLOSE SHADY LANE"
280 END
```

11.4

Ejercicio

1. Escriba programas con los que el usuario pueda añadir registros a los programas creados en el ejercicio 11.1.

11.5

.Proyectos

1. Un banco local necesita un programa que realice las siguientes funciones:

   1. Crear un archivo de clientes en un disco cuyos registros contengan los siguientes campos:

Campo 1: Apellido.
Campo 2: Nombre.
Campo 3: Domicilio.
Campo 4: Número de teléfono.
Campo 5: Número de cuenta.
Campo 6: Cantidad total de dinero en la cuenta.

2. Poner al día el campo 6 sumándole los intereses ganados en un determinado período de tiempo.
3. Escribir el registro de cualquier cliente si se introduce el número de cuenta.

2. La señora Celso tiene acciones de diversas compañías. Escriba un programa BASIC que realice las siguientes tareas:

1. Almacene datos en un disco cuyos registros tengan los siguientes campos:

   Campo 1: Nombre de la compañía.
   Campo 2: Número de acciones poseídas.
   Campo 3: Precio pagado.
   Campo 4: Dividendos recibidos.
   Campo 5: Beneficios/Pérdidas si se vende el paquete de acciones.

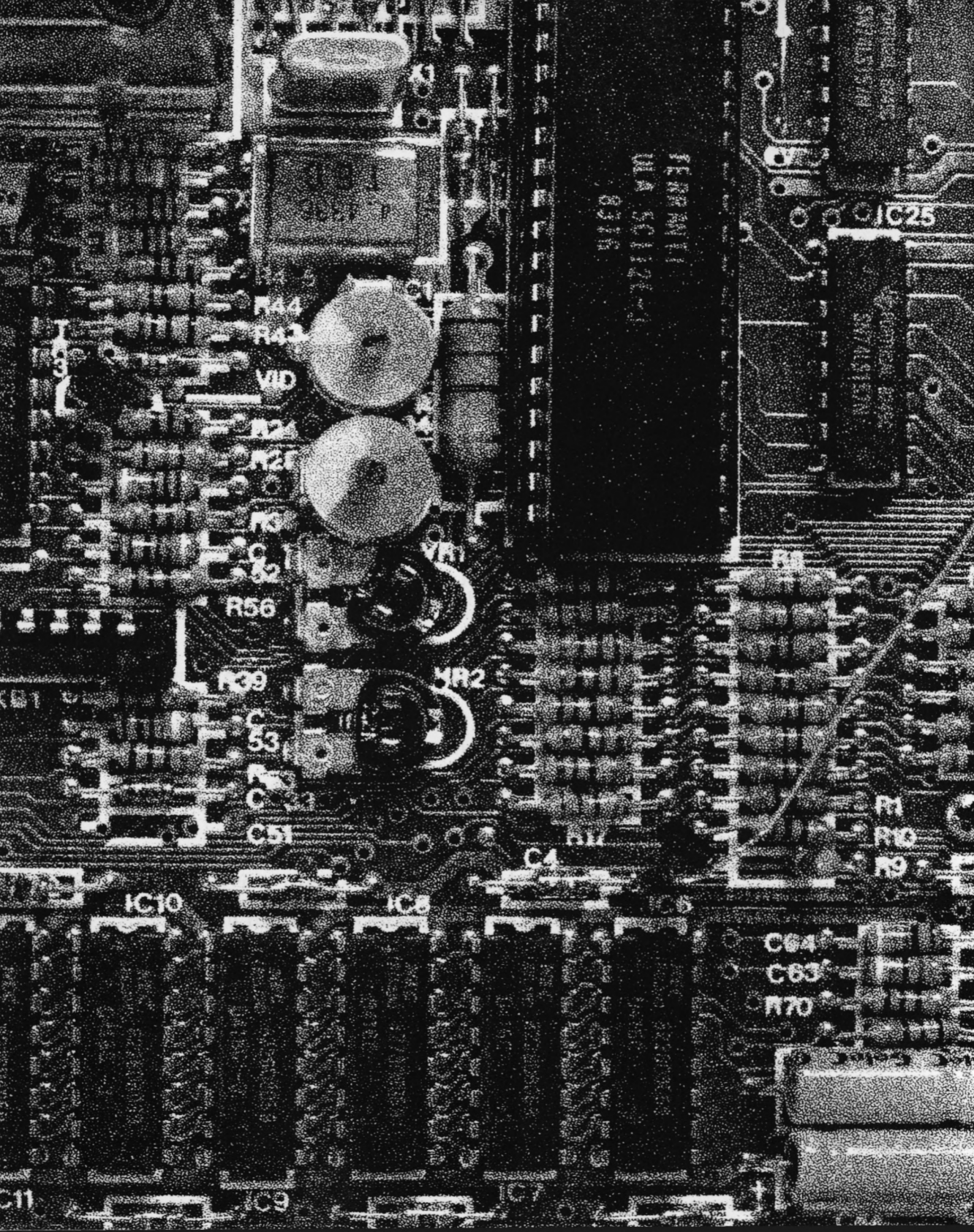
2. Escriba en pantalla los archivos.
3. Ordene las filas según el campo 1.
4. Calcule los beneficios obtenidos por la venta de acciones.

3. El profesor Fermi necesita un programa BASIC que almacene en un disco un archivo con la siguiente información de las clases que imparte:

— Materia.
— Aula.
— Nombre de los alumnos.
— Notas de los ejercicios parciales de los alumnos.
— Notas del examen final de cada alumno.
— Nota definitiva de cada alumno.

Escriba un programa controlado por menú que realice las siguientes tareas:

1. Crear el archivo.
2. Escribir los registros de todos los alumnos de una clase determinada.
3. Buscar en todo el archivo a un determinado alumno y escribir su registro.


IC25
R44
VID
R56
R39
C51
C4
R1
R9
IC10
IC8
C64
C63
R70
IC9
IC7

12
Características especiales del BASIC Applesoft

12.0
Introducción

En este capítulo trataremos muchas de las características particulares de BASIC Applesoft que lo convierten en una poderosa versión del lenguaje BASIC. Estas ventajas se pueden dividir en cinco categorías:

1. PEEK, POKE y CALL.
2. Presentación en la escritura.
3. Instrucciones gráficas.
4. Control de mandos externos y altavoz.
5. Miscelánea.

12.1
PEEK, POKE y CALL

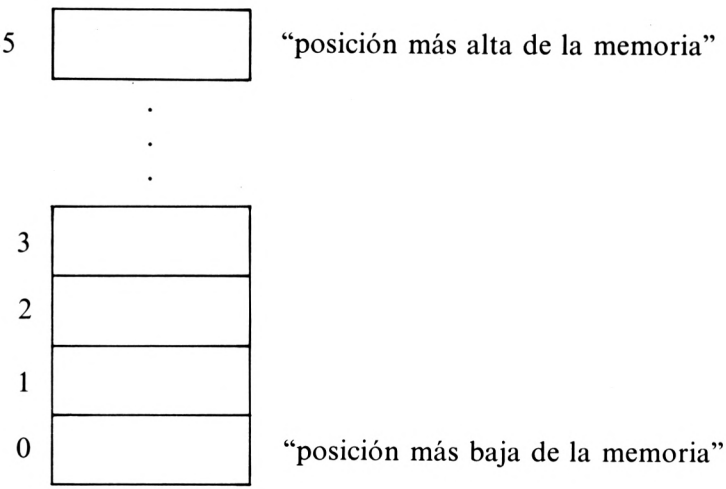
Las instrucciones PEEK, POKE y CALL son ventanas al invisible mundo de la memoria del ordenador. La verdadera importancia de estas

instrucciones se verá en la sección dedicada a la estética en la escritura, la cual no querrá perderse por nada de este mundo.

Hagamos un paréntesis para hablar brevemente de la memoria del ordenador, lo cual nos será muy útil para entender las funciones PEEK y POKE.

A) LA MEMORIA DEL ORDENADOR

Imagine una larga fila de cajas numeradas en orden empezando por cero. La memoria del ordenador es una versión electrónica micro-miniaturizada de lo que está imaginando. Lo podemos representar de la siguiente manera:



Cada caja se llama *posición de memoria,* y al número se le llama *dirección* de la posición de memoria. El tamaño del microprocesador (el circuito que procesa los datos dentro del microordenador) determina la cantidad máxima de memoria que el computador puede tener. La mayoría de los microordenadores, incluido el Apple II Plus, utilizan un microprocesador de "8-bits" que puede direccionar hasta 65.536 posiciones de memoria.

Cada posición de memoria almacena el estado de ocho señales eléctricas, llamados *bits.* Algunos de estos se encuentran conectados (ON), y otros están desconectados (OFF). A los ocho bits juntos se les denomina *byte,* y representa un número en numeración binaria. El estado "00000000" representa el número 0, mientras que el "11111111" es igual a 255 en sistema decimal. No le es necesario entender los números binarios. Simplemente recuerde que cada posición de memoria contiene un byte de información, y un byte es un número decimal comprendido entre 0 y 255.

La memoria de la que hemos estado hablando tiene dos tipos ROM y RAM. ROM significa *Read Only Memory* (memoria de sólo lectura). La información que debe permanecer permanentemente en el ordenador está almacenada en ROM. De hecho, el mismo Applesoft está almacenado en ROM, ya que el Apple siempre debe recordar el lenguaje Applesoft. En la memoria del Apple II Plus, la dirección 64708 es una posición de memoria ROM. Contiene un "208". Nada será capaz de cambiarlo, aunque se apague el ordenador. En el Apple II Plus, la ROM ocupa unas 14.000 posiciones de memoria.

RAM significa *Random Access Memory* (memoria de acceso aleatorio), que es una denominación que dice bien poco. La RAM es una memoria cuyo contenido es variable. Precisamente por esto se le llama también memoria viva. Los bytes almacenados en posiciones de memoria RAM se pierden si se apaga el ordenador o si se graba otra información encima de la antigua. La memoria RAM es donde se guardan los programas cuando se están introduciendo o ejecutando. En el Apple II Plus, se pueden direccionar unas 48.000 posiciones de memorias RAM. Esto se puede expresar también como 48K, ya que K significa kilo, es decir, 1000 en el sistema métrico decimal. Para ser más exactos, en la jerga de los ordenadores K significa 1024 en vez de 1000.

B) LA FUNCION PEEK(X)

Cerrando nuestro pequeño paréntesis, tratemos la utilidad de la función PEEK en el Applesoft BASIC. PEEK es una función como las tratadas en el capítulo 2. Requiere un argumento entre paréntesis. La función PEEK(X) devuelve el número decimal contenido en la posición de memoria X. Por ejemplo:

```
LET A=PEEK(64708)
```

asigna a A el valor 208, ya que 208 es el byte contenido en la posición 64708 antes mencionada.

Ejemplo Este programa cuenta el número de posiciones de memoria entre las direcciones 4000 y 5000 que contiene el valor 0:

```
100 LET S=0
110 FOR K=4000 TO 5000
120 IF PEEK(K)=0 THEN LET S=S+1
130 NEXT K
140 PRINT "EXISTEN ";S;" POSICIONES QUE CONTIENEN
    CERO EN ESTE INTERVALO."
150 END
```

C) LA INSTRUCCION POKE

Como debe haber supuesto, lo contrario del PEEK es el POKE. POKE es la instrucción que sirve para escribir en las posiciones de memoria, como opuesto a la lectura. Dado que las posiciones ROM son imposibles de cambiar, la instrucción POKE solo es útil en la RAM. La instrucción siguiente escribe el número 25 en la posición 5000:

```
POKE 5000,25
```

Es decir, la instrucción tiene la forma:
POKE DIRECCION, DATO

Donde DIRECCION es una posición de memoria RAM en la que se debe escribir el DATO.

Ejemplo Los caracteres que aparecen en el TRC están almacenados como números en la memoria del ordenador. Todos los números que están almacenados entre las posiciones 1024 y 2047 aparecen como caracteres en algún punto de la pantalla de vídeo (para ser exactos existen 64 posiciones de memoria en este intervalo que no aparecen en pantalla). La pantalla es, por tanto, una ventana a través de la cual se ve directamente una zona de la memoria del ordenador.

Escriba la instrucción:

```
POKE 1468,193
```

Veremos aparecer la letra "A" mágicamente no muy lejos del centro de la pantalla. Lo que ha hecho es escribir un código especial que representa a la letra "A" en una muy determinada posición de memoria. Entonces el Apple lo muestra en la pantalla (el número que se corresponde con la letra "A" es 128 más el código ASCII de la letra "A". Ver sección 12.5C).

Ejemplo El siguiente programa llena las posiciones de memoria comprendidas entre la 1024 y la 2047 con el número 193. Dado que éstas son precisamente las posiciones que muestra el TRC, el efecto producido será el llenado de la pantalla con letras "A":

```
100 FOR K=1024 TO 2047
110 POKE K, 193
120 NEXT K
130 END
```

D) LA INSTRUCCION CALL

A veces, un grupo de posiciones de memoria contienen números que son en realidad instrucciones para el microprocesador. A este grupo de números se le conoce como programa en *lenguaje máquina*. Los programas en lenguaje máquina son entendidos directamente por el microprocesador del ordenador. (Por otra parte, el BASIC debe traducirse a lenguaje máquina para que el ordenador pueda entenderlo realmente.)

Existe un extenso programa contenido en la memoria ROM llamado *monitor*. Este, controla muchas de las actividades del ordenador que nosotros hemos dado por supuestas. Por ejemplo, ¿cómo sabe el cursor que debe saltar de línea después de haberse pulsado cuarenta caracteres? ¿Qué hace desplazarse al texto una línea hacia arriba cuando se llena la pantalla y no queda espacio para seguir escribiendo? La respuesta es que el programa monitor está "monitoreando" siempre al ordenador. El programa monitor contiene subrutinas en lenguaje máquina que pueden ser utilizadas desde programas BASIC. La instrucción CALL es un puente entre programas escritos en lenguaje BASIC y los escritos en lenguaje máquina.

Para ejecutar un programa en lenguaje máquina desde un programa BASIC, escriba CALL DIRECCION, donde DIRECCION es la dirección de comienzo del programa máquina. Por ejemplo: una rutina que limpia la pantalla empieza en la posición 64600. Es parte del programa monitor. Para limpiar la pantalla escribe:

```
CALL 64600
```

Ejemplo El siguiente programa hace ascender diez líneas al texto en pantalla. La rutina del monitor que produce este efecto empieza en la posición 64624:

```
100 FOR K=1 TO 10
110 CALL 64624
120 NEXT K
130 END
```

Unas últimas palabras antes de ver las aplicaciones en presentación de textos. Las posiciones altas de memoria están expresadas como números negativos en la mayoría de los libros. Por ejemplo, la dirección 64600 es equivalente a la −936. El ordenador entiende las dos formas. Para hallar el correspondiente número negativo, reste el número 65536 de la dirección. Nosotros usaremos también el convenio de los números negativos.

Ejemplo CALL −936 es equivalente a CALL 64600
porque 64600 −65536 = −936

CALL −912 es equivalente a CALL 64624
porque 64624 −65536 = −912

12.2

Estética en la escritura

En esta sección estudiaremos varias instrucciones especiales del Applesoft que le ayudarán a escribir en pantalla con buena presentación. Las instrucciones de las que nos ocuparemos son:

A) HOME.
B) HTAB y VTAB.
C) INVERSE, FLASH y NORMAL.
D) SPEED=
E) Ventanas de texto.
F) Llamadas a subrutinas máquina mediante instrucciones CALL.

Estas instrucciones de presentación son útiles sólo en pantalla. Sus efectos en una impresora son impredecibles.

A) HOME

La instrucción HOME simplemente limpia la pantalla y sitúa el cursor en la esquina superior izquierda. HOME puede ser utilizado como una instrucción de ejecución inmediata o como una sentencia dentro de un programa BASIC. Es equivalente a CALL −936, que hemos tratado antes.

B) HTAB y VTAB

Estas dos instrucciones sirven para situar el cursor en cualquier punto de la pantalla. HTAB quiere decir "tabulador horizontal" y la instrucción:

HTAB X

hará que el cursor se mueva de la actual posición a la columna X de la misma línea. El número X debe estar comprendido entre 1 y 40. La instrucción de "tabulación vertical":

VTAB Y

mueve el cursor a la línea Y empezando a contar desde arriba de la pantalla. Lás líneas están numeradas de la 1 a la 24, empezando desde arriba. El número Y debe estar incluido en este intervalo. VTAB Y no afectará la posición horizontal del cursor.

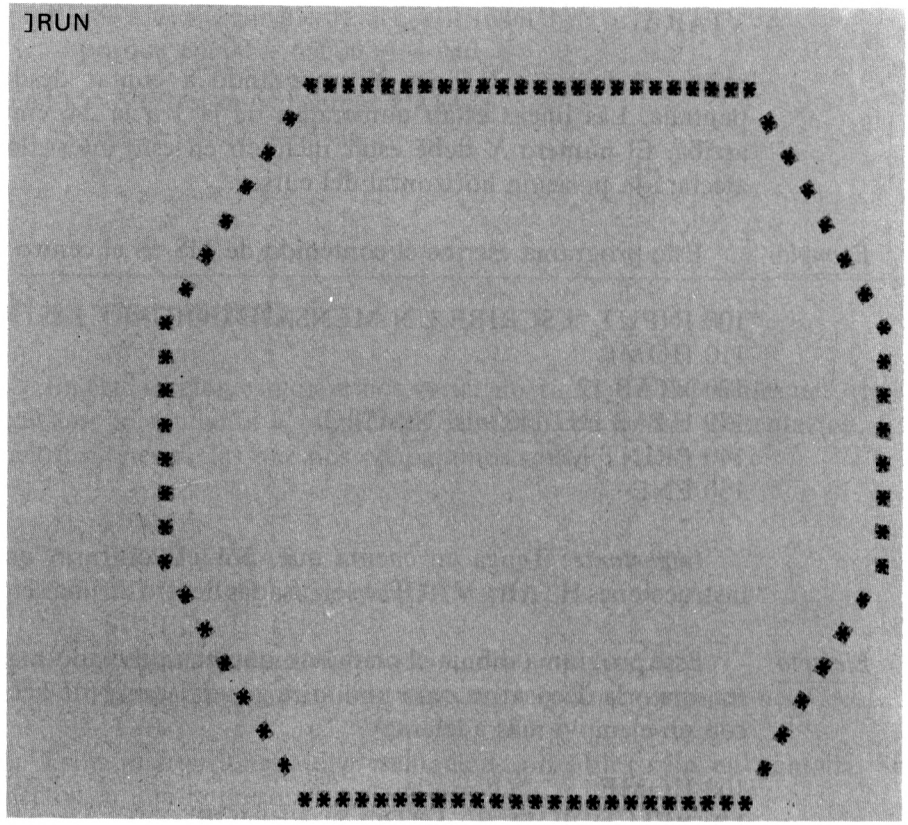
Ejemplo Este programa escribe el contenido de M$ en el centro de la pantalla:

```
100 INPUT "ESCRIBE UN MENSAJE CORTO: ";M$
110 HOME
120 VTAB 12
130 HTAB INT((40 - LEN(M$))/2)
140 PRINT M$
150 END
```

Importante: Tenga en cuenta que, por el contrario que TAB(X), las instrucciones HTAB y VTAB *no* son usadas dentro de una sentencia PRINT.

Ejemplo Este programa dibuja el octógono que hay dibujado más abajo. Puede ser una orla decorativa para un índice general, ¿no? (de hecho, se utilizará con un ejemplo más adelante).

```
100 HOME
110 REM DIBUJA EL LADO SUPERIOR
120 VTAB 2 :HTAB 10: PRINT "**********************"
130 REM DIBUJA LOS LADOS SUPERIORES INCLINADOS
140 FOR K = 3 TO 8
150 VTAB K :HTAB 12 - K :PRINT "*"
160 VTAB K :HTAB 29 + K :PRINT "*"
170 NEXT K
180 REM DIBUJA LOS LADOS VERTICALES
190 FOR K = 9 TO 15
200 VTAB K :HTAB 4 :PRINT "*"
210 VTAB K :HTAB 37 :PRINT "*"
215 NEXT K
220 REM DIBUJA LOS LADOS INCLINADOS INFERIORES
230 FOR K = 16 TO 21
240 VTAB K :HTAB K + 12 :PRINT "*"
250 VTAB K :HTAB 53 - K :PRINT "*"
260 NEXT K
270 REM DIBUJA EL LADO INFERIOR
280 VTAB 22 :HTAB 10 :PRINT "**********************"
290 END
```



Ejemplo A continuación describimos una interesante aplicación de VTAB y HTAB. Se pueden realizar dos o más inputs situando sus mensajes en la misma línea. En este programa, el usuario es preguntado por su nombre y edad. Los mensajes "ESCRIBA SU NOMBRE" y "EDAD" aparecerán en la misma línea.

```
100 HOME
110 VTAB 5
120 INPUT "ESCRIBA SU NOMBRE: ";N$
130 VTAB 5 :HTAB 30
140 INPUT "EDAD: ";A
150 END
```

```
]RUN
ESCRIBA SU NOMBRE: JUAN PEREZ      EDAD: 47
```

Como es natural, el usuario, Juan Pérez, debe pulsar RETURN después de haber escrito su nombre para que el mensaje del segundo INPUT sea escrito en pantalla.

Una versión mejorada del último ejemplo utiliza la función PEEK. La posición de memoria 37 almacena la posición vertical del cursor en cada momento. Por tanto, la instrucción VTAB PEEK(37) mantendrá el cursor en la última línea en la que se haya escrito. Esto evita tener que utilizar la función VTAB para fijar una línea determinada donde deba aparecer el mensaje del INPUT. El programa anterior se escribirá ahora así:

```
100 INPUT "ESCRIBA SU NOMBRE: ";N$
110 VTAB PEEK(37) :HTAB 30
120 INPUT "EDAD: ";A
130 END
```

La ventaja consiste en que se pueden eliminar las sentencias VTAB 5.

C) INVERSE, FLASH y NORMAL

Normalmente el Apple escribe el texto con letras blancas sobre fondo negro. Este es el modo NORMAL. La instrucción INVERSE cambia la salida de tal forma que los caracteres se escriban en negro sobre fondo blanco. La instrucción FLASH escribe caracteres que continuamente cambian de NORMAL a INVERSE y viceversa, o lo que es lo mismo: hacen FLASH. El modo FLASH es particularmente útil cuando la atención del usuario debe ser dirigida a un punto determinado de la pantalla. NORMAL vuelve a situar la salida en su modo NORMAL. Después de ejecutar una sentencia NORMAL, INVERSE o FLASH, toda la salida será escrita en el modo elegido hasta que se ejecute una de las dos restantes sentencias. Tenga en cuenta que cuando se elige un modo, éste solo afecta a la salida subsiguiente. El texto que ya está escrito en pantalla no se verá afectado.

Observación: Tenga en cuenta que si se abusa de FLASH e INVERSE la visión de la pantalla se vuelve incómoda, causando molestia a los ojos. Sea prudente con su uso.

Ejemplo He aquí un índice general en el que el título está escrito en inverso y el mensaje de salida en flash. Las opciones del índice general están escritas normalmente:

```
290 REM UNIR ESTE PROGRAMA AL ANTERIOR
295 REM PARA HACER UN INDICE GENERAL
300 VTAB 8 :HTAB 11
310 INVERSE
320 PRINT "TALONARIO ELECTRONICO"
```

```
330 NORMAL
340 PRINT
350 HTAB 12 :PRINT "1. HACER UN CHEQUE"
360 HTAB 12 :PRINT "2. HACER UN INGRESO"
370 HTAB 12 :PRINT "3. MOSTRAR EL BALANCE"
380 HTAB 12 :PRINT "4. ACABAR"
390 PRINT
400 HTAB 11
410 FLASH
420 INPUT "CUAL ES SU ELECCION? ";E
430 NORMAL
...(Seguiría el programa)...
```

No se olvide de volver al modo NORMAL, como en las líneas 330 y 430.

D) SPEED=

Seguramente se habrá dado cuenta que el Apple escribe en pantalla a gran velocidad. De hecho, a velocidad máxima el Apple puede escribir hasta 1.000 caracteres por segundo. Si usted pudiera leer a esta velocidad, leería 12.000 palabras por minuto. Afortunadamente existe un método de variar la velocidad de escritura en pantalla. La instrucción es:

SPEED=X

donde X es cualquier número entero comprendido entre el 0 y el 255. El número 255 representa la máxima velocidad (1000 cps). A medida que el número decrece también lo hace la velocidad. Cuando X=0, el Apple escribe a paso de tortuga, aproximadamente seis caracteres por segundo. Es un ritmo de lectura de aproximadamente setenta y dos palabras por minuto. La instrucción SPEED= puede ser usada en modo inmediato o diferido.

Ejemplo Escriba el siguiente programa:

```
100 SPEED=0
110 PRINT "ESTO ES A 6 CARACTERES POR SEGUNDO."
120 SPEED=100
130 PRINT "ESTO ES A 16 CARACTERES POR SEGUNDO."
140 SPEED=200
150 PRINT "ESTO ES A 100 CARACTERES POR SEGUNDO."
160 SPEED=255
```

```
170 PRINT "ESTO ES A 1000 CARACTERES POR SEGUNDO."
180 END
```

Observe que doblando el número en la instrucción SPEED= no se dobla la velocidad de escritura.

E) VENTANAS DE TEXTO

Una "ventana de texto" es una porción rectangular de la pantalla dentro de la cual puede escribirse. La ventana normal de texto es, por supuesto, toda la pantalla. Veinticuatro líneas por cuarenta columnas. ¡Pero no tiene por qué ser siempre así!

Una ventana de texto más pequeña que la pantalla da a ésta un aspecto interesante. La porción de pantalla situada fuera de la ventana aparece congelada, mientras que la actividad normal de escritura continúa dentro de la ventana. Una aplicación típica de esto puede ser la escritura de las reglas de un juego o instrucciones para el usuario, que deben permanecer siempre en pantalla. O en otra aplicación, las cabeceras de columna pueden permanecer fijas mientras que los datos que hay debajo cambian.

Variar la ventana de texto consiste en escribir cuatro números directamente en memoria mediante instrucciones POKE. Los cuatro números definen el nuevo margen izquierdo, el ancho de línea, el margen superior y el margen inferior de la ventana. Estos números van en las direcciones 32, 33, 34 y 35, respectivamente. A continuación damos los detalles:

POKE 32, I

Esta instrucción fija el margen izquierdo de la ventana de texto a I, estando I comprendido entre 0 y 39. El valor cero fija el margen totalmente a la izquierda de la pantalla, mientras que 39 fija el valor totalmente a la derecha, quedando una única columna para escribir. Piense que I es el número de columnas que quedan congeladas a la izquierda de la ventana. *Atención:* No cambie el margen izquierdo de la pantalla a no ser que también cambie el ancho de la ventana (véase lo siguiente).

POKE 33, A

Esta instrucción fija el ancho de la ventana a A, que debe ser un número comprendido entre 1 y 40. Si va a cambiar el margen izquierdo de la ventana, cambie *antes* la anchura de la ventana, para que ésta quede

definida dentro de la pantalla. Por ejemplo, si desea mover el margen izquierdo de manera que queden congeladas cinco columnas a la izquierda (POKE 32,5), entonces debe cambiar previamente la anchura de la ventana a 35 o menos (POKE 33,35). Si no se hace esto, cualquier programa BASIC que haya en ese momento en memoria puede perderse.

POKE 34, S

Esta instrucción fija el margen superior de la ventana de texto. El número S debe estar comprendido entre 0 y 23. Piense en S como el número de líneas que quedan congeladas encima de la ventana. El valor cero fija el margen en la línea superior de la pantalla, mientras que el 23 deja una línea libre en el fondo de la pantalla para escribir. Asegúrese de que el margen superior no esté fijado por debajo del margen inferior.

POKE 35, B

Esta instrucción fija el margen inferior de la ventana de texto a la línea B. El número B debe estar comprendido entre 0 y 24, donde 24 representa el fondo de la pantalla. Asegúrese de que el margen inferior no es mayor o igual que el margen superior.

Para volver a fijar la ventana de texto a su valor inicial, toda la pantalla, se pueden volver a escribir en las posiciones 32, 33, 34 y 35 los valores 0, 40, 0 y 24, respectivamente. Pero la instrucción TEXT lo hace por nosotros automáticamente. Use TEXT cuando ya no sea necesaria su propia ventana de texto.

Ejemplo Este programa escribe una tabla de conversión entre grados Fahrenheit a Celsius. Las cabeceras de columnas se congelan en la parte superior de la pantalla bajando el margen superior de la pantalla.

```
100 HOME
110 PRINT TAB(10);"FAHRENHEIT";TAB(25);"CELSIUS"
120 PRINT TAB(10);"----------          ";TAB(25);"--------"
125 REM TRES LINEAS "CONGELADAS" ARRIBA
130 POKE 34,3
140 HOME :REM ESTO NOS SITUA DENTRO DE LA VENTANA
    DE TEXTO
150 FOR F=32 TO 212
160 LET C=5/9 * (F-32)
170 PRINT TAB(14);F;TAB(25);C
```

```
180 NEXT F
190 REM YA ESTA ACABADA LA TABLA; VOLVEMOS A
    PANTALLA COMPLETA
200 TEXT
210 END
```

Ejemplo ¿Recuerda un ejemplo anterior en el que dibujábamos un octógono de asteriscos? Las siguientes líneas añadidas a aquel programa cambian la ventana de texto a una incluida dentro del octógono. Este permanecerá siempre en pantalla.

```
290 REM CAMBIO DEL MARGEN SUPERIOR PARA CONGELAR
    CINCO LINEAS SOBRE LA VENTANA
300 POKE 34,5
305 REM
310 REM CAMBIO DEL MARGEN INFERIOR PARA CONGELAR
    SEIS LINEAS BAJO LA VENTANA
320 POKE 35,18
325 REM
330 REM CAMBIO DE LA ANCHURA DE VENTANA
    A 26 COLUMNAS
340 POKE 33,26
345 REM
350 REM CAMBIO DEL MARGEN IZQUIERDO PARA
    CONGELAR 7 COLUMNAS A LA IZQUIERDA
    DE LA VENTANA
360 POKE 32,7
365 REM
370 REM AHORA NOS SITUAMOS EN EL INTERIOR DE LA
    VENTANA
380 HOME
...(El programa seguiría a continuación)...
```

De los anteriores ejemplos habrá podido sacar la conclusión de que la instrucción HOME sirve para saltar dentro de la ventana una vez que ésta ha sido definida. Unicamente limpia la parte de pantalla que cae dentro de la ventana.

F) LLAMADAS A LENGUAJE MAQUINA MEDIANTE CALL

En la última parte de la sección dedicada a la presentación en la escritura tratábamos las rutinas en lenguaje máquina que ya están escritas en el ordenador y que afectan a la representación en pantalla. Estas

rutinas pertenecen al programa monitor, ya comentado antes en este capítulo. Pueden ser ejecutadas desde dentro de su programa BASIC usando la instrucción CALL X, donde X es la dirección de memoria donde comienza la rutina (recuerde que si X es un número negativo, éste es igual a la posición de memoria menos 65536).

CALL – 958

Esta instrucción limpia la ventana de texto desde la actual posición del cursor hasta el margen inferior. Los caracteres a la izquierda del cursor y encima de éste permanecen inalterados.

CALL – 868

Esta instrucción limpia la actual línea desde el cursor hasta el final.

CALL – 912

Como ya hemos visto, esta instrucción hace que todo el texto escrito en la ventana se desplace una línea para arriba, desapareciendo la superior.

Ejemplo

El programa que sigue pide al usuario que escriba un código con la forma XX-XXX. Si el número escrito no es de longitud correcta, o si falta el guión, la entrada es rechazada y debe volverse a introducir. Estudie el programa. ¿Para qué sirve la línea 130?

```
100 PRINT "INTRODUZCA UN CODIGO (XX-XXX) "
110 VTAB PEEK(37)
120 HTAB 30
130 CALL-868
140 INPUT "";C$
150 IF LEN(C$)<>6 OR MID$(C$,3,1)<>"-" THEN GOTO 110
160 END
```

He aquí una explicación. Si el usuario escribe un número equivocado la sentencia IF-THEN en 150 envía el control a la línea 110. Esto produce una tabulación vertical a la última línea en la que se haya escrito. A continuación el cursor se mueve a la columna 30. La instrucción CALL – 868 limpia la antigua entrada errónea a partir del cursor y hasta el final de la línea. Esto deja al usuario con una línea limpia donde escribir un nuevo código.

12.3

Gráficas

Una de las más excitantes características del Apple II Plus es su capacidad para realizar gráficas en color. Existen dos tipos de gráficas, las de alta y las de baja resolución, llamadas a veces, para abreviar, "hi-res" y "lo-res", respectivamente (del inglés *high resolution* y *low resolution*)

A) GRAFICAS DE BAJA RESOLUCION

Utilizando instrucciones especiales, particulares del Applesoft, es posible "encender" pequeños rectángulos en la pantalla. En baja resolución, la pantalla está dividida en 1600 rectángulos, dispuestos en una cuadrícula de 40 por 40. Cada *pixel* (como se llaman cada uno de estos rectángulos) puede ser "iluminado" en dieciséis colores distintos. En Applesoft existen cinco instrucciones de baja resolución. Estas son: GR, COLOR=, PLOT, HLIN y VLIN. Se pueden utilizar en modo inmediato y diferido.

1. *GR*

Esta instrucción pone el Apple en modo gráfico de baja resolución. Convierte las 20 primeras líneas de la página de texto en una pantalla gráfica de 40 líneas y la pone toda a negro. Sólo se podrá escribir texto en las cuatro últimas líneas de la pantalla. Para salir del modo gráfico utilice la instrucción TEXT.

2. *COLOR=*

Esta instrucción fija el color en el que se va a dibujar. Pero no se "enciende" ningún pixel hasta que no se utilice la instrucción PLOT (véase el punto 3). Existen dieciséis colores, numerados del 0 al 15. Por ejemplo, la instrucción

```
COLOR=9
```

fija el color en naranja para posteriores dibujos. He aquí una lista de los 16 colores (sí, es verdad que hay dos grises). En una televisión en blanco y negro o en un monitor de vídeo, los colores se dibujan en distintos tonos de gris:

0 negro	4 verde oscuro	8 marrón	12 verde
1 magenta	5 gris	9 naranja	13 amarillo
2 azul oscuro	6 azul medio	10 gris	14 agua
3 púrpura	7 azul claro	11 rosa	15 blanco

3. *PLOT C,R*

La instrucción PLOT C,R rellena un pixel con el color elegido por la instrucción COLOR=. El pixel rellenado es el situado en la columna C y en la fila R. Existen cuarenta columnas numeradas del 0 al 39, siendo la 0 la que está situada más a la izquierda en la pantalla (las columnas son verticales, las filas horizontales). Las filas también están numeradas de 0 a 39, siendo la 0 la superior.

Ejemplo Escriba las siguientes instrucciones:

```
GR
COLOR=9
PLOT 0,0
PLOT 39,0
COLOR=13
PLOT 20,39
```

Podrá ver un rectángulo naranja en la esquina superior izquierda, otro en la superior derecha y un pixel amarillo en el centro de la línea inferior de la pantalla (20 columnas adelante, 39 filas abajo). Note que el color elegido se mantiene hasta que lo varía una nueva instrucción COLOR=.

Ejemplo Este programa rellena los 1600 pixels con colores escogidos al azar.

```
100 REM PASAR A MODO GRAFICO
110 GR
120 REM ESTE BUCLE VARIA LAS COLUMNAS
130 FOR C=0 TO 39
140 REM
150 REM ESTE BUCLE VARIA LAS LINEAS
160 FOR R=0 TO 39
170 REM
180 REM SE ASIGNA AL COLOR UN NUMERO ENTRE 0 Y 15
190 COLOR=INT(16*RND(1))
200 REM
210 REM SE RELLENA EL PIXEL CORRESPONDIENTE
    A LAS ACTUALES FILA Y COLUMNA
220 PLOT C,R
230 REM
240 NEXT R
250 REM
260 NEXT C
270 END
```

4. *HLIN C1,C2 AT R*

HLIN significa "línea horizontal". La instrucción HLIN C1,C2 AT R dibuja una línea horizontal en la fila R, entre las columnas C1 y C2. El número C1 debe ser menor o igual que C2. Por ejemplo:

HLIN 0,39 AT 20

dibuja una línea en el centro de la pantalla gráfica (fila 20) desde la columna 0 hasta la 39. La instrucción

HLIN 31,36 AT 5

dibuja una línea corta (seis pixels) desde la columna 31 hasta la 36 en la línea 5. Esto está más o menos por la parte superior derecha de la pantalla.

5. *VLIN R1,R2 AT C*

Esta instrucción dibuja una línea vertical desde la fila R1 hasta R2 en la columna C. El número R1 debe ser menor o igual que R2. Por ejemplo:

VLIN 0,39 AT 20

dibuja una línea vertical en el centro de la pantalla (columna 20) de arriba (fila 0) hasta abajo (fila 39). (Hablando con propiedad, no existe ninguna columna en el medio de la pantalla, porque no existe un número impar de éstas. Pero la 20 se aproxima. Lo mismo ocurre con las filas.)

Ejemplo He aquí un efecto deslumbrante. Pruébelo. Es un bucle infinito, así que cuando se canse pulse RESET o CTRL-C. Pulsando la tecla de RESET se cancelará el modo gráfico, mientras que utilizando la de CTRL-C esto no ocurrirá.

```
100 GR
110 FOR K=0 TO 39
120 COLOR=INT(16*RND(1))
130 HLIN 0,39 AT K
140 VLIN 0,39 AT K
150 NEXT K
160 GOTO 110
170 END
```

Ejemplo En este programa se dibuja un diagrama de barras para conocer el coste mensual de la calefacción de una hipotética familia. Las longitudes

de las barras verticales se toman con relación a la más alta, que representa al mes de mayor consumo. La altura de esta barra se fija arbitrariamente a tres cuartos de la altura total de la pantalla.

```
100 REM LOS SIGUIENTES DATOS SON LOS COSTES
    MENSUALES DE
110 REM CALEFACCION DISPUESTOS DE ENERO
    A DICIEMBRE
120 DATA 105, 95, 85, 70, 40, 40, 35
130 DATA 30, 35, 48, 70, 90
140 REM
150 REM ALMACENAMIENTO DE LOS DATOS
    EN UNA MATRIZ
160 DIM C(12)
170 FOR K=1 TO 12
180 READ C(K)
190 NEXT K
200 REM
210 REM BUSCA EL MAYOR VALOR
220 LET L=C(1)
230 FOR K=2 TO 12
240 IF C(K)>L THEN LET L=C(K)
250 NEXT K
260 REM
270 REM PASAMOS A MODO GRAFICO
280 REM Y DIBUJAMOS UNA BARRA POR CADA MES.
290 REM LA BARRA MAS LARGA ES DE 30 PIXELS
300 REM Y LAS DEMAS SON PROPORCIONALES.
305 GR
308 COLOR=15
310 FOR K=1 TO 12
315 REM PROPORCIONAMOS C(K) RESPECTO A 30 UNIDADES
320 LET H=C(K)/L*30
321 REM SE REDONDEA H AL ENTERO MAS CERCANO
322 LET H=INT (H+.5)
323 REM DESDE LA FILA 40-H HASTA LA 39 HAY H PIXELS
325 REM SE COMPRUEBA QUE H NO SEA CERO;
    SI NO RESULTARIA ERROR
330 IF H<>0 THEN VLIN 40-H,39 AT 3*K
340 NEXT K
350 REM SE ESCRIBEN LOS NOMBRES DE LOS MESES
    DEBAJO DE LA GRAFICA
360 HOME
```

```
365 VTAB 21
370 FOR K=1 TO 12
380 PRINT TAB(3*K+1);K;
390 NEXT K
395 PRINT:PRINT
410 PRINT TAB(10);"GASTOS MENSUALES DE CALEFACCION";
420 REM EL PUNTO Y COMA FINAL EVITA QUE
    EL TEXTO SUBA UNA LINEA
430 REM DE MAS CUANDO ACABA EL PROGRAMA.
440 END
```

B) GRAFICAS DE ALTA RESOLUCION

En alta resolución, la pantalla está dividida en 280 columnas y 160 filas, con cuatro líneas de texto debajo. La ventana de texto puede ser eliminada opcionalmente, lo cual incrementa el número de líneas a 192. Esto hace un total de 53.760 pixels. ¡Es una diferencia importante respecto a la baja resolución! Además, hay menos instrucciones que aprender. Estas son: HGR, HGR2, HCOLOR=, y HPLOT.

1. *HGR* y *HGR2*

Cualquiera de las dos instrucciones fija el ordenador en modo gráfico de alta resolución y limpia la pantalla dejándola totalmente a negro. Con HGR se mantienen cuatro líneas de texto debajo de la pantalla, igual que en baja resolución. HGR2 elimina la ventana de texto y produce gráficas en la totalidad de la pantalla.

2. *HCOLOR=*

Con la instrucción HCOLOR= se pueden asignar colores, numerados del 0 al 7. Aunque, dado que el blanco y el negro están duplicados, sólo es posible utilizar seis colores diferentes:

0 negro	3 blanco	6 rojo
1 violeta	4 negro	7 blanco
2 verde	5 azul	

3. *HPLOT C,R y variaciones*

Para "encender" un pixel en la pantalla de alta resolución, se usa la instrucción HPLOT C,R. C es un número de columna comprendido entre el 0 y el 279. En gráficas a toda pantalla (HGR2) el número R es una fila comprendida entre 0 y 191. Si existe ventana de texto, entonces R no debe ser mayor de 159.

No existen instrucciones especiales para dibujar líneas horizontales o verticales en alta resolución. En su lugar, una variante de la instrucción HPLOT dibuja una línea entre dos puntos dados. La instrucción

```
HPLOT C1,R1 TO C2,R2
```

hace que se dibuje una línea entre el punto C1,R1 y el C2,R2. Esto es mucho más versátil que las instrucciones HLIN y VLIN en baja resolución.

Ejemplo Este programa rellena totalmente la pantalla con cuatro bandas verticales de color:

```
100 HGR2
110 HCOLOR=1
120 FOR K=0 TO 69
130 HPLOT K,0 TO K,191
140 NEXT K
150 HCOLOR=2
160 FOR K=70 TO 139
170 HPLOT K,0 TO K,191
180 NEXT K
190 HCOLOR=5
200 FOR K=140 TO 209
210 HPLOT K,0 TO K,191
220 NEXT K
230 HCOLOR=6
240 FOR K=210 TO 279
250 HPLOT K,0 TO K,191
260 NEXT K
270 END
```

Para volver a modo de texto, escriba TEXT.

Ejemplo Este programa realiza un interesante dibujo en la esquina superior derecha:

```
100 HGR
110 HCOLOR=3
120 FOR K=0 TO 159 STEP 7
130 HPLOT 0,159-K TO K,0
140 NEXT K
150 END
```

Se puede utilizar también otra variante de la instrucción HPLOT. Tiene la siguiente forma:

```
HPLOT TO C,R
```

que hace que se dibuje una línea desde el último punto que se ha "encendido" hasta el C,R. Tiene distintas aplicaciones, como por ejemplo representar la actividad diaria de mercados fluctuantes.

Ejemplo

Este programa traza un gráfico del precio del oro por onza a lo largo de un período de doce semanas. También se dibuja una línea horizontal que representa la media del precio del oro a lo largo del período de tres meses antes citado. La gráfica está hecha a escala de manera que el máximo de la pantalla representa el mayor valor obtenido, y el mínimo de la pantalla el menor valor obtenido durante dicho período de tiempo. (*Nota:* Estos datos no representan ningún período real de tiempo.)

```
100 REM ALMACENA LOS VALORES DEL ORO
    EN UNA MATRIZ
105 REM Y CALCULA EL VALOR MEDIO
110 DATA 12 :REM NUMERO DE DATOS
120 DATA 285, 300, 310, 308, 318, 313
130 DATA 310, 302, 320, 333, 347
140 READ N
150 DIM G(N)
155 LET T=0
160 FOR K=1 TO N
170 READ G(K)
175 LET T=T+G(K)
180 NEXT K
185 LET AVE=T/N
190 REM
200 REM BUSCA LOS NUMEROS MAYOR Y MENOR
210 LET L=G(1) :LET S=G(1)
220 FOR K=1 TO N
230 IF G(K)>L THEN LET L=G(K)
240 IF G(K)<S THEN LET S=G(K)
250 NEXT K
260 REM
270 REM DIBUJA EL PRIMER PUNTO, A ESCALA
272 HGR
275 HCOLOR=3
280 LET Y=159/(L-S)*(L-G(1))
```

```
290 HPLOT 0,Y
300 REM
310 REM AHORA DIBUJA LOS DEMAS PUNTOS
320 FOR K=2 TO N
330 LET Y=159/(L-S)*(L-G(K))
340 REM W ES LA SEPARACION HORIZONTAL
    ENTRE LOS PUNTOS DATO
345 LET W=279/N
350 HPLOT TO W*K,Y
360 NEXT K
370 REM
430 REM DIBUJO DE LA LINEA QUE REPRESENTA
    LA MEDIA DEL PRECIO DEL ORO
440 LET Y=159/(L-S)*(L-AVE)
450 HCOLOR=6
460 HPLOT 0,Y TO 279,Y
470 REM
480 REM TEXTO AL FINAL
490 HOME
500 VTAB 21
510 PRINT "EL PRECIO MEDIO DE LAS DOCE SEMANAS
    ES $";INT(AVE+.5); " POR ONZA"
520 END
```

12.4

Control de los mandos exteriores* y el altavoz

Los mandos exteriores para juegos y el altavoz incorporado añaden una nueva dimensión al Apple II Plus que lo diferencia de otros microordenadores. Los mandos son dispositivos de entrada, y el altavoz de salida. Le mostraremos cómo puede utilizar estas posibilidades en su programa.

A) LOS MANDOS Y LOS PULSADORES

Normalmente, un mando de juegos es una cajita de plástico con un botón que puede girar en ambos sentidos. La posición del botón determina un número comprendido entre 0 y 255 que puede leer el ordenador.

* Es frecuente el uso del vocablo inglés "paddle" (remo) para designar a estos mandos, aun en castellano *(N. del T.)*.

Además, tiene de uno a tres conmutadores de dos posiciones, cuyos estados pueden ser detectados por el ordenador. Para leer el valor de los mandos desde el Applesoft existe la función PDL(X). Los pulsadores son leídos hallando el valor de determinadas posiciones de memoria mediante la función PEEK.

Concretamente, la función PDL(X) devuelve un número comprendido entre 0 y 255, dependiendo de la posición del mando X. Puede haber como máximo cuatro mandos conectados a la vez a un mismo Apple, que están numerados 0, 1, 2 y 3. Así que X debe ser un número comprendido entre 0 y 3. (*Nota:* La mayoría de fabricantes de mandos para juegos los venden por pares. Cuando están conectados al Apple estos son el mando 1 y el 0. Dado que los mandos 2 y 3 deben ser conectados en el mismo sitio, éstos no pueden ser usados si ya están conectados los mandos 1 y 0. Se deben usar adaptadores especiales si se quieren utilizar los cuatro mandos a la vez.)

Ejemplo Este programa muestra la posición de los mandos:

```
100 PRINT PDL(0),PDL(1)
110 GOTO 100
120 END
```

Pruébelo. Verá un flujo de números escritos en la pantalla a dos columnas. Mueva el mando O en el sentido contrahorario. Los números de la columna de la izquierda se aproximarán a cero. Gírelo en sentido horario. Los números se aproximarán a 255. Lo mismo ocurrirá con los números de la otra columna si gira el mando 1. Pare el bucle infinito con RESET o CTRL-C.

Ejemplo Este programa sirve para dibujar gráficas de baja resolución con los mandos. Estos mueven un cuadro blanco por la pantalla que va dejando un rastro según se mueve:

```
100 GR
110 COLOR = 15
120 REM EL MANDO O INDICA LA COLUMNA
130 REM PROPORCIONADO DE LOS NUMEROS 0 A 39
140 LET C = INT(PDL(0)/255 * 39)
150 REM EL MANDO 1 INDICA LA FILA, TAMBIEN
    PROPORCIONADA
160 LET R = INT(PDL(1)/255 * 39)
170 PLOT C,R
180 GOTO 140
190 END
```

De nuevo esto es un lazo infinito, pero como veremos más adelante, se puede utilizar un pulsador para salir del bucle y parar el programa.

Incorporaremos ahora los pulsadores a los programas anteriores. Para ver si el botón O está pulsado, se utiliza la instrucción PEEK(−16287). El valor devuelto será mayor que 127 sólo si el pulsador está apretado. Las direcciones −16286 y −16285 tienen la misma función, sólo que referido a los pulsadores 1 y 2. Por ejemplo, la sentencia

```
IF PEEK(-16287)>127 THEN PRINT
   "EL PULSADOR 0 ESTA APRETADO"
```

escribirá el mensaje sólo si el pulsador está apretado.

Ejemplo Este programa muestra las posiciones de los mandos y el estado de los botones:

```
100 TEXT :HOME
110 PRINT "MANDO 0";TAB(12);"1";TAB(15);"PULSADOR 0";
    TAB(30);"1"
115 REM SE FIJA EL MARGEN SUPERIOR POR DEBAJO
    DE LA CABECERA
120 POKE 34,2
125 HOME :REM SALTAMOS DENTRO DE LA VENTANA
130 LET P0=PDL(0)
140 LET P1=PDL(1)
150 LET B0$="OFF"
160 LET B1$="OFF"
170 IF PEEK(-16287)>127 THEN B0$="ON"
180 IF PEEK(-16286)>127 THEN B1$="ON"
190 PRINT TAB(8);P0;TAB(12);P1;TAB(22);B0$;TAB(30);B1$
200 GOTO 130
210 END
```

Ejemplo El programa de dibujo anterior puede mejorarse utilizando los pulsadores para cambiar los colores y salir del programa. He aquí la versión mejorada:

```
100 GR
110 COLOR=15
115 REM SI SE PULSA EL BOTON O ELIGE UN COLOR
    AL AZAR
118 IF PEEK(-16287)>127 THEN COLOR=INT(16*RND(1))
120 REM EL MANDO 0 INDICA EL NUMERO DE COLUMNA
```

```
130 REM PROPORCIONADA A UN NUMERO ENTRE 0 Y 39
140 LET C=INT(PDL(0)/255 * 39)
150 REM EL MANDO 1 INDICA EL NUMERO DE FILAS;
    TAMBIEN PROPORCIONADA.
160 LET R=INT(PDL(1)/255 * 39)
170 PLOT C,R
180 REM SI SE APRIETA EL BOTON 1, PARA EL PROGRAMA
190 IF PEEK(-16389)>127 THEN GOTO 210
200 GOTO 115
210 END
```

B) USO DEL ALTAVOZ

La posición de memoria −16336 es la clave para la generación de sonidos con el altavoz de Apple. Haciendo un PEEK a esta posición, el altavoz es excitado, produciendo un "clic" audible. La manera más sencilla de que el altavoz pueda producir tonos consiste en hacer el PEEK a esta posición de memoria lo suficientemente rápido como para generar una frecuencia audible. Este corto programa produce un zumbido:

```
100 LET X=PEEK(-16336)
110 GOTO 100
120 END
```

Para conseguir un sonido más refinado es necesario "apretar" más el lazo, de manera que se haga el PEEK más veces por segundo. Pero el bucle anterior es lo máximo que podemos conseguir en BASIC. La solución al problema consiste en realizar el bucle en lenguaje máquina, que se ejecuta mucho más rápido que en Applesoft.

He aquí una subrutina que puede utilizar en sus programas. Lo que hace es: mediante instrucciones POKE introduce en la memoria un programa que produce un tono de frecuencia y duración variable.

```
63997 POKE 770,173 :POKE 771,48 :POKE 772,192:
      POKE 773,136 :POKE 774,208 :POKE 775,4 :
      POKE 776,198 :POKE 777,1 :POKE 778,240

63998 POKE 779,8 :POKE 780,202 :POKE 781,208 :
      POKE 782,246 :POKE 783,166 :POKE 784,0 :
      POKE 785,76 :POKE 786,2 :POKE 787,3 :
      POKE 788,96

63999 RETURN
```

Para utilizar la subrutina, escríbala en su programa y escriba la línea.

```
10 GOSUB 63997
```

Los números comprendidos entre 1 y 255 representan la nota y la duración del tono. Su programa debe realizar unos POKE a las direcciones 0 y 1 con los valores de la nota y la duración, respectivamente. Una vez hecho esto, la instrucción CALL 770 ejecuta la rutina del lenguaje máquina, que produce el tono.

Ejemplo Este pequeño programa le permite escribir los números correspondientes a nota y duración, y acto seguido oír el sonido que resulta. Para parar el programa introduzca ceros en los dos valores.

```
  10 GOSUB 63997
 100 INPUT "¿DURACION? ";D
 120 INPUT "¿NOTA? ";P
 130 IF D=0 AND P=0 THEN GOTO 180
 140 POKE 0,P
 150 POKE 1,D
 160 CALL 770
 170 GOTO 100
 180 END
63997 POKE 770,173 :POKE 771,48 :POKE 772,192 :
      POKE 773,136 :POKE 774,208 :POKE 775,4 :
      POKE 776,198 :POKE 777,1 :POKE 778,240
63998 POKE 779,8 :POKE 780,202 :POKE 781,208 :
      POKE 782,246 :POKE 783,166 :POKE 784,0 :
      POKE 785,76 :POKE 786,2 :POKE 787,3 :
      POKE 788,96
63999 RETURN
```

Ejemplo He aquí el comienzo de la canción "Twinkle, Twinkle, Little Star":

```
 10 GOSUB 63997
 20 REM CADA DOS NUMEROS SON NOTA Y DURACION
 30 DATA 120,100,120,100,80,100,80,100
 40 DATA 70,100,70,100,80,160,90,100,90,100
 50 DATA 95,100,95,100,106,100,106,100,120,160
 60 FOR K=1 TO 14
 70 READ P,D
 80 POKE 0,P
 90 POKE 1,D
100 CALL 770
```

```
  110 NEXT K
  120 END
63997 POKE 770,173 :POKE 771,48 :POKE 772,192 :
      POKE 773,136 :POKE 774,208 :POKE 775,4 :
      POKE 776,198 :POKE 777,1 :POKE 778,240
63998 POKE 779,8 :POKE 780,202 :POKE 781,208 :
      POKE 782,246 :POKE 783,166 :POKE 784,0 :
      POKE 785,76 :POKE 786,2 :POKE 787,3 :
      POKE 788,96
63999 RETURN
```

Hay probablemente una fórmula que relaciona los números de las notas con las verdaderas notas de cada sonido. ¡Pero no hemos sido capaces de encontrarla! Los números de las notas de la pequeña melodía anterior fueron encontrados por experimentación.

12.5

Miscelánea

A pesar de que en este momento su conocimiento del Applesoft BASIC es prácticamente total, aún quedan algunas funciones y sentencias que merecen su atención. Por razones de continuidad y claridad, estos temas fueron dejados aparte en capítulos anteriores, aunque no entrañan ningún tipo de dificultad.

Aquí se incluyen:

A) La sentencia DEF FN
B) La sentencia GET
C) La función ASC(X$)
D) La función CHR$(X)

A) LA SENTENCIA DEF FN

Como sabemos, el BASIC es suministrado con una biblioteca de funciones ya definida que realizan tareas especiales, como generar números aleatorios o calcular raíces cuadradas. Además de estas funciones ya definidas, el Applesoft tiene un método para que el programador pueda crear sus propias funciones. Estas funciones se conocen como *funciones definidas por el usuario.* Pueden servir para redondear números, elegir números aleatorios entre 1 y 6, calcular el tabulado necesario para centrar un título, y cualquier otra cosa que se le pueda ocurrir.

La sentencia DEF FN es la clave para hacer funciones definidas por el usuario. He aquí un típico ejemplo:

```
100 DEF FN C(X)=5/9*(X-32)
```

Podemos reconocer la parte derecha de la expresión como la fórmula para convertir grados Fahrenheit en grados Celsius o centígrados. Una vez que la expresión anterior ha sido situada en el programa, la expresión FN C(X) devuelve los grados centígrados equivalentes a X grados Fahrenheit, como ocurre en el siguiente programa:

```
100 DEF FN C(X)=5/9*(X-32)
110 FOR F=68 TO 86 STEP 9
120 PRINT F, FN C(F)
130 NEXT F
140 END
```

```
]RUN
68              20
77              25
86              30
```

La sentencia DEF FN requiere un nombre y una fórmula para la función que se está definiendo. Cualquier nombre válido para una variable numérica puede ser utilizado como nombre de función. El nombre dado a la función anterior es C, y la fórmula es 5/9*(X−32). A continuación del nombre debe haber unos paréntesis que contengan una variable numérica conocida como variable simbólica. Esta variable simbólica se usa únicamente para relacionar la fórmula con el nombre de la función, de tal manera que los valores puedan ser "pasados a la fórmula". La variable simbólica puede ser cualquier variable numérica. Por ejemplo, las tres siguientes sentencias definen exactamente la misma función:

```
100 DEF FN C(X)=5/9*(X-32)
ó
100 DEF FN C(A)=5/9*(A-32)
ó
100 DEF FN C(F)=5/9*(F-32)
```

Una vez que la función ha sido definida en una sentencia DEF FN ésta es referenciada en el programa por FN, seguido por el nombre y un argumento, por ejemplo FN C(5). *Importante:* la sentencia DF FN debe ser ejecutada antes de cualquier referencia a la función que define.

Ejemplo He aquí un programa que emplea una función definida por el usuario para redondear números a la centena más cercana. Téngase en cuenta que se pueden dar nombres descriptivos a las funciones, siempre y cuando no se utilicen palabras reservadas:

```
10 REM ESTE PROGRAMA MUESTRA EL INTERES GANADO
20 REM Y EL NUEVO CAPITAL AL FINAL DE CADA AÑO
30 REM DE UNA INVERSION DE 100000 PTAS
40 REM AL 8% ANUAL DURANTE CINCO AÑOS
45 REM
50 DEF FN ROUND(X)=INT(100*X+.5)/100
55 REM EL CAPITAL EMPIEZA A 100000 PTAS
60 LET P=100000
65 PRINT "AÑO";TAB(10);"INTERESES";TAB(20);"CAPITAL"
70 FOR A=1 TO 5
75 REM CALCULO DEL INTERES EN UN AÑO
80 LET I=.08*P
85 REM CALCULO DEL NUEVO CAPITAL
90 LET P=P+1
100 REM ESCRIBE LOS RESULTADOS REDONDEADOS
    A CENTIMOS
110 PRINT A;TAB(10);FN ROUND(I);TAB(20);FN ROUND(P)
120 NEXT A
130 END
```

```
]RUN

AÑO       INTERES     CAPITAL
1         8,000       108,000
2         8,640       116,640
3         9,331       125,971
4         10,078      136,049
5         10,884      146,933
```

Ejemplo Para generar números enteros aleatorios entre 1 y 6 se puede utilizar la siguiente función definida por el usuario:

```
10 DEF FN RDM(X)=INT(6*RND(1))+1
```

Téngase en cuenta que la fórmula de la derecha no necesita de una variable para definirse. De todas maneras, la variable simbólica *debe* aparecer en la definición de la función. Cuando se utiliza la definición, en el programa debe aparecer un argumento numérico entre los paréntesis de la función, a pesar de que éste no sirve para nada. Por ejemplo:

```
PRINT FN RDM(0)
PRINT FN RDM(12.2)
PRINT FN RDM(A)
```

imprimirán un número entero entre 1 y 6

Ejemplo Esta función determina a qué distancia se debe tabular de tal manera que una cadena quede centrada en pantalla:

```
100 DEF FN T(X)=INT(20-LEN(A$)/2)
```

Importante: La variable simbólica *debe* ser numérica; no puede ser una variable tipo cadena. La única manera de que esta función pueda trabajar sobre una cadena es si la variable tipo cadena que contiene la cadena que debe ser centrada, es parte de la fórmula. Este programa centra las cadenas F$ y G$, que deben ser asignadas a A$ antes de ser empleadas por la función:

```
100 DEF FN T(X)=INT(20-LEN(A$)/2)
110 LET F$="CAPITULO 12"
120 LET G$="CARACTERISTICAS ESPECIALES DEL
    APPLESOFT BASIC"
130 REM
140 REM SE ASIGNA F$ A A$ PARA USARLO EN LA FUNCION
150 LET A$=F$
160 PRINT TAB(FN T(0));A$
170 REM ASIGNACION DE G$ A A$
180 LET A$=G$
190 PRINT TAB (FN T(0));A$
200 END
```

Tenga en cuenta otra vez que el argumento de la función es irrelevante, así que hemos utilizado el cero.

B) LA SENTENCIA GET

La sentencia GET es una nueva forma de sentencia de asignación que, como INPUT, espera a que el usuario escriba un dato en el teclado. La instrucción:

```
GET A$
```

toma la primera tecla pulsada en el teclado y se la asigna a A$. No requiere que sea pulsada la tecla RETURN, y no imprime el carácter pulsado.

Pruebe a ejecutar el siguiente programa:

```
100 PRINT "EMPIEZA A ESCRIBIR, PULSA 'Q' PARA ACABAR"
110 GET A$
120 PRINT A$
130 IF A$ = "Q" THEN GOTO 150
140 GOTO 110
150 END
```

Encontrará que a medida que usted escribe las letras, éstas aparecen instantáneamente en la pantalla. No es necesario pulsar RETURN. Si quitamos la línea 120 del programa, no aparecerá nada en pantalla mientras ejecuta el programa. Pero la sentencia GET sigue todavía cogiendo cada carácter que usted escriba.

GET puede ser usada también con una variable numérica, por ejemplo:

```
GET X
```

pero no se lo recomendamos. Si se pulsa una letra en vez de un número aparecerá el mensaje SYNTAX ERROR y el programa se parará. Una forma mejor de tomar un número del teclado es:

```
100 GET X$
20 LET X = VAL(X$)
```

Si X$ es un carácter distinto de 0, 1, 2, ..., 9, la función VAL devolverá el valor 0 y el programa no se parará.

Ejemplo

En este programa, la sentencia GET es usada para tomar la elección del usuario en un índice general:

```
100 REM MENU DE ELECCIONES
110 PRINT "1. ELECCION UNO..."
120 PRINT "2. ELECCION DOS..."
130 PRINT "3. ELECCION TRES..."
140 PRINT "4. ELECCION CUATRO..."
150 PRINT
160 PRINT "PULSE UN NUMERO, NO PULSE RETURN:";
170 GET N$
175 PRINT N$
180 LET N = VAL (N$)
190 ON N GOSUB 1000, 2000, 3000, 4000
200 GOTO 110
...(continúa el programa)...
```

Una desventaja de usar la función GET en un índice general es que éste no puede tener más de nueve elecciones, si están numeradas empezando con uno. Además, la sentencia GET puede ser frustrante para un usuario acostumbrado a pulsar RETURN después de introducir los datos.

La sentencia INPUT no permite que sean aceptadas comas o puntos y comas como parte de una cadena, a no ser que el usuario preceda la cadena con un signo de admiración. En el ejemplo que sigue, hay una subrutina BASIC que usando el GET evita la presencia del signo de interrogación precedente. La subrutina coge caracteres uno a uno mientras son escritos y los concatena para crear una línea completa de entrada.

Las comas, puntos y comas, y hasta signos de interrogación no serán rechazados.

Ejemplo

Estudie con mucho cuidado la subrutina que empieza en la línea 1000. La línea 1040 utiliza la función ASC que trataremos brevemente más adelante. Por ahora, nos basta con saber que la línea 1040 está comprobando si se ha pulsado la tecla RETURN.

```
100 REM ESTE PROGRAMA UTILIZA GET EN LUGAR
    DE INPUT
110 REM PARA PODER ACEPTAR COMAS Y PUNTOS Y COMAS
120 REM COMO PARTE DE LA ENTRADA
130 REM
140 PRINT "INTRODUZCA CUAL ES SU FECHA DE
    NACIMIENTO>> ";
150 GOSUB 1000
160 LET F$=A$
170 PRINT "INTRODUZCA CUAL ES LA HORA DE SU
    NACIMIENTO>> ";
175 GOSUB 1000
180 LET T$=A$
190 PRINT
200 PRINT "USTED NACIO EL ";F$;" A LAS ";H$
210 GOTO 9999
220 REM
1000 REM SUBRUTINA PARA TOMAR LA ENTRADA
1005 REM EN PRINCIPIO SE INICIALIZA A$ COMO
     LA CADENA NULA
1010 LET A$=""
1015 REM TOMA UN CARACTER DEL TECLADO
1020 GET B$
1030 PRINT F$
1040 REM SI EL CARACTER FUE RETURN ENTONCES ACABAR
```

```
1050 IF ASC(B$)=13 THEN GOTO 1090
1060 REM EL CARACTER PULSADO SE CONCATENA A A$
1070 LET A$=A$+B$
1080 GOTO 1020
1090 RETURN
9999 END
```

```
]RUN

   INTRODUZCA SU FECHA DE NACIMIENTO>> DICIEMBRE 2, 1960
   INTRODUZCA LA HORA DE SU NACIMIENTO>> 7:45 AM

USTED NACIO EN DICIEMBRE 2, 1960 A LAS 7:45 AM
```

Recuerde, los caracteres subrayados fueron escritos por el usuario.

Un defecto de la subrutina del ejemplo anterior es que en caso de error no hay posibilidad de corrección. Todos los caracteres, incluso los incorrectos, son concatenados A$ en la línea 1070. En seguida corregiremos este defecto.

C) LA FUNCION ASC(X$)

Cada carácter del teclado tiene asignado un número único. Cuando se pulsa una tecla, el número asociado con ésta se almacena en memoria. El Apple, como la mayoría de los microordenadores, asigna números a los caracteres, de acuerdo con el código ASCII. (ASCII: "American Standard Code for Information Interchange", Código Normalizado Americano para el Intercambio de Información.) En el código ASCII, por ejemplo, a la letra A le corresponde el código 65. De hecho, todas las letras, de la A a la Z, tienen asignados los códigos del 65 al 90, respectivamente. Incluso las teclas tales como RETURN y las flechas tienen asignado su propio código. He aquí una lista de los códigos ASCII de la mayoría de los caracteres del teclado del Apple.

CODIGOS ASCII DE LOS CARACTERES

Carácter	*Código*	*Carácter*	*Código*
Campanilla (CTRL-G)	7	0	48
		1	49
Flecha a la izquierda	8	2	50
		3	51
		4	52

Carácter	*Código*	*Carácter*	*Código*
RETURN	13	5	53
		6	54
Flecha a la derecha	21	7	55
		8	56
Espacio	32	9	57
		:	58
!	33	;	59
"	34	<	60
#	35	=	61
$	36	>	62
%	37	?	63
&	38	@	64
'	39		
(	40	A	65
)	41	B	66
*	42	C	67
+	43	.	.
,	44	.	.
–	45	.	.
.	46	Y	89
/	47	Z	90

La función ASC(X$) devuelve el código ASCII del primer carácter de la cadena contenido en X$. Por ejemplo:

```
LET M = ASC("ABC")
```

asigna a M el valor 65, ya que 65 es el código ASCII de la letra A.

Ejemplo Este programa escribe los códigos ASCII de las teclas pulsadas en el teclado:

```
100 PRINT "EMPIEZA A ESCRIBIR, PULSAR 'Q' PARA ACABAR"
110 GET A$
120 PRINT A$, ASC(A$)
130 IF A$ = "Q" THEN GOTO 150
140 GOTO 110
150 END
```

Ejemplo He aquí una mejora a la subrutina de un ejemplo anterior que permitía entradas con comas y puntos y comas. En esta rutina las teclas pulsadas por error pueden ser borradas utilizando la tecla correspondiente

a la flecha. La línea 1080 comprueba el código ASCII de la tecla pulsada, y si éste es el correspondiente a la flecha izquierda corrige el error.

```
100 PRINT "ESCRIBA ALGO."
105 PRINT "UTILIZA LA FLECHA A LA IZQUIERDA
    PARA CORREGIR ERRORES."
110 GOSUB 1000
120 PRINT "ESTO ES LO QUE ESCRIBIO:"
130 PRINT A$
140 GOTO 9999
150 REM
1000 REM SUBRUTINA PARA TOMAR LA ENTRADA
1010 REM PRIMERO SE INICIALIZA A$ A LA CADENA NULA
1020 LET A$=""
1030 REM TOMA UN CARACTER DEL TECLADO
1040 GET B$
1050 REM SI EL CARACTER ERA RETURN ENTONCES ACABAR
1060 IF ASC(B$)=13 THEN GOTO 1180
1070 REM SI EL CARACTER ERA FLECHA A LA IZQUIERDA
    ENTONCES ELIMINA EL ULTIMO CARACTER
1080 IF ASC(B$)=8 THEN GOTO 1120
1090 REM EL CARACTER ES CONCATENADO A A$
1100 LET A$=A$+B$
1110 GOTO 1160
1120 REM SE ELIMINA UN CARACTER DE A$
1130 REM EXCEPTO SI NO HAY CARACTERES QUE ELIMINAR!
1140 IF LEN (A$)=0 THEN GOTO 1040
1150 LET A$=MID$(A$,1,LEN(A$)-1)
1160 PRINT B$;
1170 GOTO 1040
1180 PRINT
1190 RETURN
9999 END
```

D) LA FUNCION CHR$(X)

Al contrario que la función ASC(X$), que devuelve el código ASCII del primer carácter de una cadena, CHR$(X) devuelve el carácter cuyo código ASCII es X. Por tanto:

```
PRINT CHR$(65)
```

escribirá una A en pantalla.

Se puede utilizar CHR$ para escribir caracteres especiales. Por ejemplo:

```
PRINT CHR$(7); "DESPIERTA!"
```

producirá un *beep* en el altavoz.

Existen tres caracteres que el Apple es capaz de escribir en pantalla, pero que no están en el teclado. Estos son el paréntesis cuadrado izquierdo "[", la barra inclinada a la izquierda "\" y el subrayado "_". Sus códigos ASCII son, respectivamente, 91, 92 y 95. Este programa escribe la cadena APPLE][centrada en la pantalla:

```
100 LET A$ = "APPLE ]" + CHR$(91)
110 LET T = INT(20 - LEN(A$)/2)
120 PRINT TAB(T);A$
130 END
```

Por cierto, el paréntesis cuadrado a la derecha, "]", se escribe pulsando "shift-M". Lo que ocurre es que, por alguna extraña razón, el símbolo no está grabado en la tecla.

Apéndice A: Cómo opera el Apple II Plus

El Apple II Plus puede utilizarse de dos maneras, con o sin el sistema operativo de discos, llamado normalmente DOS (Disk Operative System). El DOS es un programa escrito en lenguaje máquina, no en BASIC, que permite al ordenador "hablar" a la unidad de discos. Si usted quiere grabar programas para emplearlos posteriormente, o usar programas que han sido previamente grabados, entonces es necesario el DOS, pero si no se requiere el almacenamiento permanente, no hace falta dicho programa.

FUNCIONAMIENTO DEL APPLE SIN EL DOS

Esta es la forma más fácil. Siéntese frente al ordenador. En la parte trasera de éste encontrará el interruptor de alimentación. Conéctelo, y verá que se encienden dos luces. Una es la luz que está en la esquina inferior izquierda del teclado, y en ella figura la palabra *power*. La otra está situada en el controlador de discos. Esta se enciende siempre que el disco está girando, que es lo que estará haciendo ahora mismo. Pulse la tecla de RESET. El disco se parará y la luz se apagará. Conecte el TRC (que es el televisor o el monitor de vídeo). En la pantalla se verá un paréntesis cuadrado a la derecha (]), y el cursor, que es un rectángulo intermitente. Ahora está usted preparado para programar en BASIC Apple-

soft. Sin embargo, cualquier programa que usted escriba no puede ser grabado en un diskette porque no hay DOS.

FUNCIONAMIENTO DEL APPLE CON EL DOS

Supongamos por un momento que usted tiene en sus manos un diskette "inicializado". Cualquier diskette inicializado sirve. Si usted, por el momento, no dispone de ninguno, siga leyendo. Aprenderá cómo preparar uno.

Primero unas palabras sobre la manipulación de diskettes. Es fácil que se deterioren, no los doble, ni los raye, ni los mutile. Y, especialmente, no les acerque imanes. También debe saber que se pueden estropear con el calor. No los deje en guanteras de coches o sobre proyectores. No toque nunca el disco de plástico recubierto de una película magnética, que está situada entre las dos cartulinas negras.

¿Dónde estábamos? Ah, sí, usted tenía un diskette inicializado en la mano. Abra la unidad de disco. Con la etiqueta mirando hacia arriba y la abertura oval alejada lo más posible de usted, inserte el diskette y cierre la puerta. Ahora conecte la corriente como se describió antes. La luz roja se apagará ella sola después de que el disco haya girado algunos segundos. Lo que ocurre en estos segundos es que el programa DOS está siendo cargado del diskette al ordenador automáticamente. La expresión que se utiliza en inglés para este proceso es *booting* que se puede traducir por "arrancar" o "cebar". Acaba ahora mismo de *cebar el sistema.* [*Nota:* Ocasionalmente el disco puede seguir girando más tiempo del debido. Algo va mal. Pulse RESET y retire el disco. Puede ser debido a que: 1) el disco no esté inicializado; 2) que esté inicializado con una versión anterior del DOS; 3) que el diskette no estuviera bien situado dentro del controlador, o 4) que el controlador necesita reparación. Si usted sabe que el diskette está inicializado con la versión correcta del DOS (llamada DOS 3.3), entonces intente inicializar el sistema de nuevo.]

Después de que se haya apagado la luz roja, deberá ver el cursor en pantalla. En este momento, está usted preparado para programar en Applesoft. Además es capaz de cargar y grabar programas. Debe conocer alguna de las instrucciones más importantes del DOS, como son: INIT, CATALOG, LOAD, SAVE, y DELETE. Siga leyendo.

1. INIT

Probablemente quiera tener un diskette personal en el cual guardar sus propios programas. Los diskettes se pueden comprar en los establecimientos del ramo por unos cientos de pesetas cada uno. Con uno bastará

para un semestre de trabajo, pero quizá a usted le gustaría tener otro donde guardar los programas importantes. Cuando usted compra un diskette éste está en blanco, no sólo no tiene ningún programa, sino que tampoco tiene el DOS. Para poner el DOS en un diskette en blanco usted debe inicializarlo. Inicializando el diskette automáticamente escribe el programa DOS en él. He aquí el procedimiento:

a) Primero arranque el sistema con cualquier diskette ya inicializado.

b) Saque el diskette inicializado y ponga en su lugar el que está en blanco. Cierre la puerta del controlador.

c) Escriba NEW y pulse RETURN para dejar espacio en blanco donde trabajar.

d) Escriba un programa corto tal como

```
10 HOME
20 PRINT "ESTE DISKETTE PERTENECE A JUAN PEREZ"
30 END
```

e) Sin ningún número delante escriba INIT HOLA y pulse la tecla RETURN.

El disco girará aproximadamente durante un minuto y cuando se pare ya podrá utilizar su disco para arrancar el sistema, porque ya tiene una copia del DOS en él. También tiene un programa en BASIC llamado HOLA ¿Se lo suponía...? El programa HOLA es el que usted acaba de escribir justo antes de inicializar el diskette. El programa HOLA se ejecutará automáticamente cada vez que usted arranque el sistema con este disco. Intente rearrancar el sistema ahora. ¿Se acuerda cómo? Simplemente apague el ordenador y vuelva a encenderlo con su diskette dentro del controlador. Pero atención, si usted inicializa un diskette que no está en blanco pierde toda la información contenida en él.

2. CATALOG

Usted puede ver los programas que hay almacenados en su disco escribiendo CATALOG. Fíjese que el primer programa se llama HOLA. ¿De dónde viene?

Una línea típica del CATALOG puede tener la siguiente forma:

```
A 006 TRATAMIENTO DE TALONARIOS
```

La letra A significa que el programa está escrito en Applesoft. El número 006 es más o menos una medida de lo largo que es el programa. Cuanto más largo sea el programa más alto será el número. Finalmente, TRATAMIENTO DE TALONARIOS es el nombre con el que está grabado el programa.

3. LOAD

Para introducir en la zona de trabajo del ordenador un programa que está almacenado en disco, escriba LOAD NOMBRE DEL PROGRAMA, donde NOMBRE DEL PROGRAMA es el nombre de cualquier programa que se pueda hallar en el catálogo. Una vez que el programa ha sido cargado, usted puede ejecutarlo o trabajar sobre él. Además, si sólo desea ejecutarlo, puede escribir RUN NOMBRE DE PROGRAMA.

4. SAVE

Los programas que usted escribe pueden ser grabados en el diskette escribiendo SAVE NOMBRE DE PROGRAMA. El nombre del programa puede ser cualquier combinación de hasta 30 caracteres, pero que debe empezar por una letra y no debe tener comas en él; los espacios y otros caracteres especiales sí sirven. Puede elegir el nombre para su programa cuando esté dispuesto a almacenarlo, no cuando empiece a escribirlo. ATENCIÓN: *Si usted graba un programa utilizando el nombre de otro programa ya existente, éste será borrado.*

5. DELETE

Los programas que no se deseen pueden ser eliminados de su diskette escribiendo DELETE NOMBRE DE PROGRAMA. Recordará la instrucción DELETE cuando intentando grabar un programa, se produzca un error del tipo DISK FULL (disco lleno).

Bien, eso es todo. Hay mucho más que aprender desde luego, pero lo que acaba usted de leer cubre todo lo fundamental.

Apéndice B: Palabras reservadas en Applesoft

En el capítulo 1 usted aprendió que ningún nombre de variable Applesoft puede contener una palabra reservada. De igual modo, inocentes combinaciones de nombres válidos de variables con ciertas instrucciones producirán inadvertidamente palabras reservadas. Por ejemplo:

```
10 IF B<A THEN GOTO 100
```

se lista como:

```
10 IF B<AT HEN GOTO 100
```

La solución es el uso de paréntesis:

```
10 IF (B<A) THEN GOTO 100
```

He aquí una lista de las combinaciones de las que uno debe procurar mantenerse alejado:

ABS	AND	ASC	AT	ATN
CALL	CHR$	CLEAR	CONT	COS
DATA	DEF	DEL	DIM	DRAW
END	EXP	FLASH	FN	FOR
FRE	GET	GOSUB	GOTO	GR
HGR	HGR2	HLIN	HOME	HPLOT
HTAB	IF	INPUT	INT	INVERSE
LEFT$	LEN	LET	LIST	LOAD
LOG	MID$	NEW	NEXT	NORMAL
NOT	NOTRACE	ON	ONERR	OR
PDL	PEEK	PLOT	POKE	POP
POS	PRINT	READ	RECALL	REM
RESTORE	RESUME	RETURN	RIGHT$	RND
RUN	SAVE	SGN	SHLOAD	SIN
SQR	STEP	STOP	STORE	STR$
TAN	TEXT	THEN	TO	TRACE
USR	VAL	VLIN	VTAB	WAIT
XPLOT	XDRAW			

Las siguientes palabras se pueden utilizar, si no van seguidas de un signo igual ("="): SCALE, SPEED, y ROT.

Las siguientes palabras se pueden utilizar si no van seguidas de paréntesis ("("): SCRN, SPC, y TAB, pero no pueden emplearse como las últimas letras del nombre de una matriz.

Las palabras HIMEM y LOMEM pueden usarse si no van seguidas por dos puntos (":"). Dado que los dos puntos son utilizados en Applesoft para separar sentencias dentro de la misma línea, es fácil que se produzcan combinaciones ilegales. Por tanto, es mejor evitar su uso.

Las palabras COLOR= y HCOLOR= son también palabras reservadas, pero como incluyen a la palabra OR no hay posibilidad de utilizarlas.

Finalmente, las palabras IN # y PR # son también palabras reservadas. De todas maneras, el signo # las invalida para usarlas como variables.

Apéndice C: Depuración de programas con y sin mensajes de error

Muy poca gente, incluso programadores profesionales, pueden escribir programas que funcionen directamente a la primera. En el léxico de los programadores los errores en los programas se llaman *bugs*, y la depuración de programas se conoce como *debugging**. Algunos errores se muestran a través de los mensajes de error que se producen durante la ejecución del programa. Otros errores, tales como los fallos lógicos, pueden producir errores que no causen mensajes. En este apéndice trataremos la técnica de encontrar errores con y sin mensajes de error. Lea el apéndice D para aprender a editar líneas de programa en las cuales sabemos que hay errores.

A) DEPURADO CON AYUDA DE MENSAJES DE ERROR

La siguiente es una lista de los mensajes de error más comunes y las probables clases que lo originan:

1. SYNTAX ERROR

Este error se produce cuando una sentencia está incorrectamente construida. Las causas principales suelen ser la falta de

* *Bug:* Garrapata en inglés. *Debugging:* Depuración *(N. del T.).*

paréntesis, la utilización de palabras reservadas, las palabras mal escritas, la puntuación incorrecta, etc.

2. UNDEF'D STATEMENT ERROR

Este error ocurre cuando se intenta hacer un salto a un número de línea que no existe en el programa.

3. BAD SUBSCRIPT ERROR

Este error ocurre cuando los subíndices de una matriz sobrepasan las dimensiones de ésta, tal como referirse a A(16) cuando A ha sido dimensionado con DIM A(15). También puede ocurrir si el número de dimensiones es incorrecto, tal como referirse a A(16) cuando la matriz ha sido dimensionada por DIM A(20,30). En un programa que no utiliza matrices, el error puede ocurrir cuando se olvide un símbolo de multiplicación ("*"), como por ejemplo en:

```
100 LET A=P(1+R)^T
```

donde debía poner LET A=P*(1+R)^T

4. ILLEGAL QUANTITY ERROR

Este error ocurre cuando el argumento de una función está fuera de rango, o cuando el subíndice de una matriz es negativo. Por ejemplo, usando SQR(−3) o LOG(0) en un programa, resultaría error ya que son funciones matemáticamente imposibles. La expresión LEFT$(A$,0) causará error porque 0 está fuera del rango correcto para la función LEFT$.

5. TYPE MISMATCH ERROR

Este error es el resultado del intento de usar un dato tipo cadena cuando se requiere un dato numérico, o viceversa. Cada una de las siguientes sentencias producirá un error del tipo TIPE MISMATCH:

```
LET A$=14
LET C=B$
PRINT LEN(X)
IF G$<P THEN GOTO 100
```

6. RETURN WITHOUT GOSUB ERROR

Este error significa que se ha alcanzado una sentencia RETURN, pero que no existe la correspondiente sentencia GOSUB

a donde volver. Típicamente éste es el resultado de no aislar las subrutinas del programa principal. Si el ordenador cae en una subrutina al final del programa principal, se producirá un error del tipo RETURN WITHOUT GOSUB al alcanzar la sentencia RETURN. Una solución normal es poner una sentencia GOTO entre el programa principal y la subrutina.

7. NEXT WITHOUT FOR ERROR

Este error ocurrirá si la sentencia NEXT muestra una variable índice distinta que la variable índice del FOR, por ejemplo en

```
10 FOR J=1 TO 10
        .
        .
        .
50 NEXT K
```

También ocurrirá si la sentencia NEXT existe pero no hay ningún FOR en el programa. Si usted decide quitar un FOR de su programa quite también la sentencia NEXT correspondiente.

8. REDIM'D ARRAY ERROR

Una vez que una matriz es dimensionada, no puede volverse a dimensionar. En caso de hacerlo aparecerá un error del tipo REDIM'D ARRAY. He aquí un ejemplo:

```
10 LET A(5)=8
20 DIM A(100)
```

La línea 10 en este ejemplo hace que la matriz A sea dimensionada a 10 automáticamente ("por defecto", en la jerga de los ordenadores) ya que es la primera referencia a la matriz. Por tanto, la línea 20 es un intento de redimensionar la matriz. Si es absolutamente necesario redimensionar la matriz, será necesario utilizar la instrucción CLEAR (se puede situar en la línea 15 del ejemplo anterior). Pero tenga cuidado porque CLEAR puede borrar los valores actuales de *todas* las variables del programa.

9. OUT OF DATA ERROR

Este error es el resultado de intentar asignar datos con una sentencia READ cuando todos los datos de sentencias DATA han sido ya asignados, o cuando no hay ningún DATA.

10. UNDEF'D FUNCTION ERROR

Este error ocurre cuando se intenta usar una función definida por el usuario (ver capítulo 12) que no ha sido previamente definido.

11. STRING TOO LONG ERROR

Este error ocurrirá si al concatenar se produce un resultado de más de 255 caracteres.

12. OUT OF MEMORY ERROR

Este error puede ocurrir por varias causas, pero la más común es el anidamiento de sentencias GOSUB con una profundidad superior a 24. No es normalmente intención del programador tener una subrutina que llame a otra subrutina, y ésta a su vez llame a otra, así hasta 24 veces, pero puede ocurrir involuntariamente, por ejemplo, cuando una sentencia GOTO devuelve el control al programa principal en lugar de un RETURN. Así, el siguiente programa produciría un error del tipo OUT OF MEMORY:

```
100 REM SUMA LOS NUMEROS DE 1 A 100
110 LET S=0
120 FOR I=1 TO 100
130 GOSUB 500
140 NEXT I
150 PRINT "LA SUMA ES ";S
160 GOTO 999
500 REM SUBRUTINA PARA ACUMULAR LOS NUMEROS
510 LET S=S+1
520 GOTO 140
999 END
```

Fíjese que la línea 520 envía el control a la 140, no con un RETURN, sino con una sentencia GOTO. La subrutina nunca es completada, sino que es llamada una y otra vez hasta que el error del tipo OUT OF MEMORY interrumpe el programa en el bucle 24. Sustituyendo la línea 520 por 520 RETURN se solucionará el problema. (En particular, este programa estaría mejor sin ningún tipo de subrutina, pero no estamos tratando aquí el tema del estilo.)

13. DIVISION BY ZERO ERROR

Al intentar dividir por cero se produce este error. La división por cero es matemáticamente imposible.

B) DEPURADO SIN LA AYUDA DE MENSAJES DE ERROR

Los programas pueden contener errores que no produzcan mensajes de error. Esto es normalmente causado por fallos en el razonamiento lógico del programa. Un síntoma de que estos errores existen, es la aparición y salidas indeseadas o inesperadas. Para ayudarle a buscar estos errores hacemos dos sugerencias:

1. Sitúe a lo largo del programa algunas sentencias PRINT que muestren el valor de algunas variables durante la ejecución del programa. Por este procedimiento puede localizarse cualquier variable que tenga valores extraños.
2. Puede situar en su programa la instrucción STOP. Cuando se está ejecutando un programa, éste se interrumpirá al alcanzar la sentencia STOP. En este punto puede utilizar instrucciones de ejecución inmediata, especialmente PRINT y LET, para mostrar o cambiar los valores de cualquier variable del programa. Escribiendo CONT se reanudará de nuevo la ejecución del programa en el punto que fue interrumpido.

```
.
.     (líneas del programa)
.
100 STOP
.
.     (líneas del programa)
.
```

```
]RUN
BREAK IN 100

]PRINT Z
14
]CONT
     (el programa sigue ejecutándose
     a partir de la línea 100)
```

Atención: Añadir o cambiar cualquier línea al programa hará imposible la continuación y todas las variables serán puestas a cero. Se producirá un error del tipo CAN'T CONTINUE (imposible continuar). No se preocupe, simplemente vuelva a reejecutar el programa.

La depuración es una habilidad que sólo se aprende con mucha práctica. Es una parte importante de la creación de un programa en sí mismo. Unas palabras de aviso: No permita que otros depuren sus propios programas, o perderá una valiosa experiencia.

Apéndice D: Edición de líneas de programa utilizando el control del cursor

Una línea de programa que contiene errores puede ser siempre corregida volviéndola a escribir. Pero hay otra manera. Se llama *control de cursor* y está relacionado con el uso de teclas especiales para mover el cursor sobre la pantalla (el cursor es el rectángulo intermitente que indica dónde va a escribirse el siguiente carácter). Editar una línea de programa usando el control de cursor supone:

1. Listado de la línea (LIST).
2. Movimiento del cursor al principio de la línea.
3. Recorrido de la línea con la flecha a la derecha.
4. Eliminar el error cuando éste se alcanza.
5. Recorrido del resto de la línea.
6. Pulsado de la tecla RETURN.

Si usted es como la mayoría de la gente, se dará cuenta que editar líneas usando el control de cursor es, en principio, difícil. Pero también se dará cuenta que es rentable en aprendizaje, ya que una vez dominado se ahorra bastante tiempo.

A) TECLAS ESPECIALES PARA CONTROLAR EL CURSOR

Hay unas teclas que hacen que el cursor se mueva por la pantalla. Le será util estar sentado frente al Apple II Plus, mientras lee esto, para que usted pueda ver por sí mismo los efectos.

La tecla ESC:

Esta tecla se pulsa una única vez. Después de haberla pulsado, las teclas I, J, K y M se transforman en las teclas que mueven el cursor. Pulsar alguna tecla distinta de éstas causa el que se desconecte el modo de control de cursor. Las posiciones relativas de estas teclas en el teclado se ven en este diagrama:

(arriba)
I

(izquierda) J K (derecha)

M
(abajo)

En este modo los caracteres sobre los que se mueve el cursor *no* son copiados en la memoria del APPLE.

La flecha a la derecha:

Esta tecla mueve el cursor a la derecha. A medida que el cursor pasa sobre los caracteres de la pantalla, estos son copiados en la memoria del Apple exactamente igual que si hubieran sido escritos sobre el teclado. Esto no es lo mismo que utilizar ESC-K, ya que éste no copia los caracteres en la memoria.

La flecha a la izquierda:

Esta tecla mueve el cursor a la izquierda. Si el cursor pasa sobre caracteres en una línea del programa, estos son borrados de la memoria del ordenador.

La tecla de repetición:

Esta tecla, cuando se pulsa, hace que otra tecla se repita automáticamente mientras esté pulsada la primera. Esto ahorra una gran cantidad de tiempo ya que la edición de una fila supone pulsar repetidamente la tecla de flecha a la derecha.

B) LECCION ILUSTRADA DE EDICION POR CONTROL DEL CURSOR

Para los ejemplos que vienen a continuación, para representar al cursor se usará el carácter de subrayado "_". *Importante:* Si usted comete errores mientras edita la línea de programa, pulso CTRL-X. Esto producirá que sea cancelada cualquier edición en curso. Liste la línea de nuevo y empiece otra vez.

Ejemplo La línea de programa

10 PRINT "VE USTED UND ERRER?"

contiene dos errores. Trabajemos primero para cambiar ERRER a ERROR

1. Liste la línea en pantalla:

   ]LIST 10

   10 PRINT "VE USTED UND ERRER?"

   ]_

   Fíjese que la posición del cursor está situada debajo del cero en el número de línea.

2. Mueva el cursor a la línea pulsando ESC una vez. Entonces pulse la tecla I dos veces, para mover el pulsor dos líneas arriba. La pantalla ahora aparece así:

   ]LIST 10

   1_0 PRINT "VE USTED UND ERRER?"

   ]

3. Ahora mueva el cursor al principio de línea pulsando J una vez. Esto es importante. No utilice la flecha a la izquierda; no funcionaría aquí. Ahora tenemos:

   ]LIST 10

   1_0 PRINT "VE USTED UND ERRER?"

   ]

4. Usando la flecha a la derecha, mueva el cursor, parándose en la letra que debe ser cambiada:

```
]LIST 10
10 PRINT "VE USTED UND ERRER?"
]
```

5. Haga la corrección y continúe con la flecha a la derecha hasta que haya rebasado la línea entera:

```
]LIST 10
10 PRINT "VE USTED UND ERROR?"_
]
```

6. Pulse RETURN, y la línea corregida será de nuevo incluida en el programa. Lístelo para estar seguro:

```
]LIST 10
10 PRINT "VE USTED UND ERROR?"
]_
```

Ejemplo Corrijamos ahora el error UND para que quede UN.

Realice los pasos 1, 2, 3 y 4 hasta mover el cursor a la letra D. La pantalla se verá así:

```
]LIST 10
10 PRINT "VE USTED UND ERROR?"
]
```

Nuestro objeto es saltar sobre la letra D. Pulse la tecla ESC una vez. Pulse K y el cursor se moverá a la derecha una posición, *sin* copiar la letra en la memoria. Usando la tecla de flecha a la derecha continúe recorriendo la línea hasta el final, y entonces pulse RETURN. Escriba LIST para ver si la corrección se ha efectuado:

```
]LIST 10
10 PRINT "VE USTED UN ERROR?"
]_
```

En el siguiente ejemplo le enseñaremos a insertar caracteres en una línea.

Ejemplo Suponga que la línea

```
10 IF X>3 THEN GOTO 200
```

es parte de su programa. Si después de haberla introducido usted decide que debe poner:

```
10 IF X>3 AND X<8 THEN GOTO 200
```

puede insertar los caracteres extra de esta manera:

1. Liste la línea y utilice ESC, I, J y la FLECHA A LA DERECHA para situar el cursor en el espacio detrás del 3 (recuerde que hay que ir al comienzo de línea utilizando la tecla J antes de empezar a moverse con la FLECHA A LA DERECHA):

```
]LIST 10
10 IF X>3_THEN GOTO 200
]
```

2. Pulse la tecla ESC y luego la I para moverse una posición para arriba:

```
]LIST 10
10 IF X>3 THEN GOTO 200
]
```

3. Ahora escriba los caracteres que quiere insertar:

```
]LIST 10
            AND X<8_
10 IF X>3 THEN GOTO 200
]
```

4. Pulse ESC y luego M para hacer volver el cursor a la línea de programa:

```
]LIST 10
            AND X<8
10 IF X>3 THEN GOTO 200
]
```

5. Puesto que ESC está todavía activo, pulse la J repetidamente hasta volver al espacio después del 3.

```
]LIST 10

            AND X<8
10 IF X>3_THEN GOTO 200

]
```

6. Ahora repase el resto de la línea utilizando la FLECHA A LA DERECHA y después pulse RETURN. Liste para ver si es correcta.

```
]LIST 10

10 IF X>3 AND X<8 THEN GOTO 200

]_
```

C) NOTA IMPORTANTE SOBRE LA EDICION DE LINEAS DE PROGRAMA LARGAS

Las líneas de programa que ocupan dos o más líneas de pantalla también pueden ser editadas utilizando el control del cursor. De todas maneras, es recomendable escribir la instrucción POKE 33,33 antes de listar la línea que va a ser editada. Esto cambia la anchura de la pantalla a 33 caracteres y tiene el deseable efecto adicional de eliminar los espacios no deseados que son insertados automáticamente por el Apple. Por ejemplo:

```
]LIST 50

50  PRINT "ESTE ES UN EJEMPLO DE UNA
    LINEA DE PROGRAMA BASTANTE LARGA.
    FIJESE EN LOS ESPACIOS EXTRA QUE
    APPLE INSERTA AL PRINCIPIO DE CADA
    LINEA DE PANTALLA."

]POKE 33,33

]LIST 50

50 PRINT "ESTE ES UN EJEMPLO DE UNA LINE
A BASTANTE LARGA: FIJESE EN LOS ESPACIOS
QUE APPLE INSERTA AL PRINCIPIO DE CADA L
INEA DE PANTALLA."

]TEXT
```

Utilice la FLECHA A LA DERECHA para repasar copiando la línea 50 listada en primer lugar y también copiará los espacios indeseados. Sin estos espacios, en la segunda línea 50, la FLECHA A LA DERECHA copiará sólo los caracteres que pertenecen a la línea de programa.

Para volver la pantalla a su modo original, con 40 caracteres de ancho, escriba TEXT en modo inmediato, como se indica arriba.

Soluciones a los ejercicios de los capítulos

Capítulo 1

Ejercicios 1.1

1. a, c, f
2. Todos son cadenas.
3. *a*) 17, *b*) 19, *c*) 14, *d*) 28

Ejercicios 1.2

1. a, b, c
2. a, c, d, f, h

Ejercicios 1.4

1. a, b, c, e
2. a, b, f

Ejercicios 1.5

1. *b*) E$ es una variable tipo cadena, no se le puede asignar el valor numérico contenido en S.
   c) Las variables no pueden empezar con un número.
   f) No se puede asignar una cadena a una variable numérica.
   h) M$ es una variable tipo cadena, pero Y es una variable numérica.
   i) F es una variable numérica, pero K$ es tipo cadena.

   Los errores producidos en *b*), *f*), *h*), *i*) se denominan errores tipo MISMATCH.

2.

	T	R	R$	Q$	T$
	0	0			
LET T=45	45	0			
LET R=22.22	45	22.22			
LET T$="MICHELLE"	45	22.22			MICHELLE
LET Q$="ESCRIBIERON"	45	22.22		ESCRIBIERON	MICHELLE
LET R$="BEATLES"	45	22.22	BEATLES	ESCRIBIERON	MICHELLE

3.

	A	B	C
	0	0	0
LET A=3	3	0	0
LET B=5	3	5	0
LET C=A	3	5	3
LET A=B	5	5	3
LET B=C	5	3	3

Ejercicios 1.6

1.

	F	G	X
	0	0	0
LET F = 3	3	0	0
LET G = 8	3	8	0
LET X = F	3	8	3
LET F = G	8	8	3
LET G = X	8	3	3

2.

	T	P	Z
	0	0	0
LET T = 7	7	0	0
LET P = 9	7	9	0
LET Z = P	7	9	9
LET P = T	7	7	9
LET T = Z	9	7	9

3.

	X$	Y$	Z1$	K$	P$
LET X$ = "DULCE"	DULCE				
LET Y$ = "BUENA"	DULCE	BUENA			
LET Z1$ = "PRINCIPE"	DULCE	BUENA	PRINCIPE		
LET K$ = "NOCHE"	DULCE	BUENA	PRINCIPE	NOCHE	
LET P$ = X$	DULCE	BUENA	PRINCIPE	NOCHE	DULCE
LET X$ = Y$	BUENA	BUENA	PRINCIPE	NOCHE	DULCE
LET Y$ = K$	BUENA	NOCHE	PRINCIPE	NOCHE	DULCE
LET K$ = Z1$	BUENA	NOCHE	PRINCIPE	PRINCIPE	DULCE
LET Z1$ = P$	BUENA	NOCHE	DULCE	PRINCIPE	DULCE
LET P$ = "!"	BUENA	NOCHE	DULCE	PRINCIPE	DULCE

4.

	A	B	C	D
	0	0	0	0
LET A = 5	5	0	0	0
LET B = 4	5	4	0	0
LET C = 3	5	4	3	0
LET D = C	5	4	3	3
LET C = B	5	4	4	3
LET B = A	5	5	4	3
LET A = D	3	5	4	3

5.

	R$	Q$	T$	K$
LET R$ = "MARY"	MARY			
LET Q$ = "JUAN"	MARY	JUAN		
LET T$ = "AMA A"	MARY	JUAN	AMA A	
LET K$ = R$	MARY	JUAN	AMA A	MARY
LET R$ = Q$	JUAN	JUAN	AMA A	MARY
LET Q$ = T$	JUAN	AMA A	AMA A	MARY
LET T$ = K$	JUAN	AMA A	MARY	MARY

Ejercicios 1.7

1.

	V	S	R$	TRC
	0	0		
PRINT V	0	0		0
LET V = 17	17	0		
PRINT V	17	0		17
LET S = 42	17	42		
PRINT S	17	42		42
LET V = S	42	42		
PRINT "V YA ESTA"	42	42		V YA ESTA
PRINT V	42	42		42
LET R$ = "ESTO ES DIVERTIDO"	42	42	ESTO ES DIVERTIDO	
PRINT R$	42	42	ESTO ES DIVERTIDO	ESTO ES DIVERTIDO
PRINT "FIN"	42	42	ESTO ES DIVERTIDO	FIN

Ejercicios 1.8

1.

	X	A	C3
	0	0	0
LET X = 4	4	0	0
LET A = X − 2	4	2	0
LET C3 = A*X	4	2	8
LET X = C3^A	64	2	8

2.

	E	F
	0	0
LET E = 3	3	0
LET F = E*E	3	9
LET F = F − E	3	6
LET F = F − E	3	3
LET E = F^E	27	3

3.

	A	TRC
	0	
LET A = 6	6	
PRINT A*3+2	6	20
PRINT A/3*2	6	4
LET A = (A − 4)^2	4	
PRINT A − A/4	4	3
PRINT (A − A)/4	4	0
LET A = A*A − A + A/A	13	
PRINT "A*A − A + A/A"	13	A*A − A + A/A

4.

	X	K	F	TRC
	0	0	0	
LET K = 5	0	5	0	
LET X = K/K	1	5	0	
LET F = K − X	1	5	4	
PRINT K + F	1	5	4	9
LET X = K + K	10	5	4	
LET K = (K − F)/K	10	0.2	4	
PRINT X*(X − K)	10	0.2	4	98

5.

	P	R	A	TRC
	0	0	0	
LET P = 2500	2500	0	0	
LET R = .15	2500	.15	0	
LET A = P*(1 + R)	2500	.15	2875	
PRINT A	2500	.15	2875	2875

6.
```
LET P = 2500
LET R = .15
LET N = 5
LET A = P*(1 + R)^N
PRINT A
```

7.

	X1	X2	X3	X4	X5	TRC
	0	0	0	0	0	
LET X1 = 1	1	0	0	0	0	
LET X2 = 2*X1	1	2	0	0	0	
LET X3 = 3*X2	1	2	6	0	0	
LET X4 = 4*X3	1	2	6	24	0	
LET X5 = 5*X4	1	2	6	24	120	
PRINT X1	1	2	6	24	120	1
PRINT X2	1	2	6	24	120	2
PRINT X3	1	2	6	24	120	6
PRINT X4	1	2	6	24	120	24
PRINT X5	1	2	6	24	120	120

8. Continuación de diagrama de cajas del ejercicio:

	X	Y	TRC
LET Y = Y + 1	6	4	
LET X = X*Y	24	4	
PRINT X	24	4	24
LET Y = Y + 1	24	5	
LET X = X*Y	120	5	
PRINT X	120	5	120

9.

	S	C1	T1	C2	T2	P	TRC
	0	0	0	0	0	0	
LET S = 250	250	0	0	0	0	0	
LET C1 = 76.25	250	7625	0	0	0	0	
LET T1 = S*C1	250	7625	1906250	0	0	0	
PRINT T1	250	7625	1906250	0	0	0	1906250
LET C2 = 81.5	250	7625	1906250	8150	0	0	
LET T2 = S*C2	250	7625	1906250	8150	2037500	0	
LET P = T2 − T1	250	7625	1906250	8150	2037500	131250	
PRINT P	250	7625	1906250	8150	2037500	131250	131250

10. Antes de PRINT P añada las instrucciones:

```
LET D=4.5*S
LET P=P+D
```

11. Antes de PRINT P añada las instrucciones:

```
LET B1=.05*T1
LET B2=.07*T2
LET P=P-B1-B2
```


Capítulo 2

Ejercicios 2.1

1. *a*) PRINT SQR(X+Y)
   RESULTADO: 2.64575131
   b) PRINT SQR(X*Y)
   RESULTADO: 3.46410162
   c) PRINT SQR(X+SQR((X+4*Y)/3))
   RESULTADO: 2.34874679
   d) PRINT SQR(X-Y)
   RESULTADO: ILLEGAL QUANTITY ERROR ya que no se puede hacer la raíz cuadrada de un número negativo.
2. LET K=3.4
   LET X1=SQR(K)
   LET X2=-SQR(K)
   PRINT X1
   PRINT X2
   RESULTADOS: 1.84390889 y -1.84390889.
   LET K=7
   LET X1=SQR(K)
   LET X2=-SQR(K)
   PRINT X1
   PRINT X2
   RESULTADOS: 2.64575131 y -2.64575131.
   LET K=150
   LET X1=SQR(K)
   LET X2=-SQR(K)
   PRINT X1
   PRINT X2
   RESULTADOS: 12.2474487 y -12.2474487.
3. LET A=130000
   LET P=100000
   LET R=SQR(A/P)-1
   PRINT R
   RESULTADO: .140175425. Que es aproximadamente un 14%.
4. LET A=1
   LET B=-7
   LET C=-11
   LET X1=(-B+SQR(B^2-4*A*C))/(2*A)
   LET X2=(-B-SQR(B^2-4*A*C))/(2*A)
   PRINT X1
   PRINT X2
   RESULTADOS: 8.32182538 y -1.32182538.
   LET A=5
   LET B=-1
   LET C=-5
   LET X1=(-B+SQR(B^2-4*A*C))/(2*A)
   LET X2=(-B-SQR(B^2-4*A*C))/(2*A)
   PRINT X1
   PRINT X2
   RESULTADOS: 1.10498756 y -.904987562.
   LET A=14
   LET B=111
   LET C=-2
   LET X1=(-B+SQR(B^2-4*A*C))/(2*A)
   LET X2=(-B-SQR(B^2-4*A*C))/(2*A)
   PRINT X1
   PRINT X2
   RESULTADOS: .0179772586 y -7.94654869.
5. LET S=(X+Y-ABS(X-Y))/2
6. LET A=INT(10*A+.5)/10
7. LET A=INT(.01*A + .5)/.01
   o LET A=INT(A/100+.5)*100
8. PRINT INT (20*RND(1))+1
   PRINT INT(20*RND(1))+1
   PRINT INT(20*RND(1))+1
   PRINT INT(20*RND(1))+1
   PRINT INT(20*RND(1))+1
9. PRINT INT(10*RND(1))+6
   PRINT INT(10*RND(1))+6
   PRINT INT(10*RND(1))+6
   PRINT INT(10*RND(1))+6
   PRINT INT(10*RND(1))+6
10. LET P1=179000000
    LET P2=203000000
    LET T2=1970-1960
    LET K=(LOG(P2)-LOG(P1))/T2
    LET T1=2000-1960

```
LET Q=P1*EXP(K*T1)
PRINT Q
  RESULTADO: .296090896.
11. LET A=1.732
LET B=1
LET C=2
LET CS=(B^2+C^2-A^2)/(2*B*C)
PRINT CS
  RESULTADO: .500044
```

NOTA: No podemos escribir LET COS(A1)= (B^2+C^2−A^2)/(2*B*C) porque sólo pueden aparecer variables a la izquierda de una sentencia LET. La expresión COS(A1) no es una variable; es una función.

Ejercicios 2.2

1. *a*) 19 *b*) VIE *c*) E *d*) RTEJO *e*) 5 *f*) TEJO *g*) YA V *h*) OR *i*) I

2. *a*) Falta $ después de MID.
b) El argumento debe ser numérico.
c) El argumento debe ser una cadena.
d) El número de caracteres debe ser mayor que cero.
e) Faltan comillas.
f) El argumento debe ser sólo una cadena.

Capítulo 3

Ejercicios 3.4

1 y **2.**

a)

	P	TRC
	0	
100 LET P=126	126	
110 PRINT "UNA PERSONA QUE PESE"	126	UNA PERSONA QUE PESE
120 PRINT P	126	126
130 PRINT "KILOS EN LA TIERRA"	126	KILOS EN LA TIERRA
140 PRINT "PESARA"	126	PESARA
150 PRINT P/6	126	21
160 PRINT "KILOS EN LA LUNA"	126	KILOS EN LA LUNA
170 END	126	

b)

	T1	T2	T3	A	TRC
	0	0	0	0	
10 LET T1=80	80	0	0	0	
20 LET T2=85	80	85	0	0	
30 LET T3=93	80	85	93	0	
40 LET A=(T1+T2+T3)/3	80	85	93	86	

	T1	T2	T3	A	TRC
50 PRINT "MI NOTA MEDIA"	80	85	93	86	MI NOTA MEDIA
60 PRINT A	80	85	93	86	
70 END	80	85	93	86	

3. *a)*

```
10 LET X=23
20 LET Q=2
30 LET L=X-Q
40 LET X=X+Q
50 PRINT X+Q
60 END
```

b)

```
100 LET E1=-6
110 LET T=1
120 LET R$="LISTO"
130 E1=E1*E1
140 PRINT R$
150 END
```

4.

	B	D	C	S	
	0	0	0	0	
100 LET B=45885	45885	0	0	0	
110 LET D=12550	45885	1255	0	0	
120 LET C=4576	45885	1255	4576	0	
130 LET S=B+D-C	45885	1255	4576	53859	
140 PRINT S	45885	1255	4576	53859	53859
150 END	45885	1255	4576	53859	

5.

	B	D	C	S	TRC
	0	0	0	0	
100 LET S=B+D-C	0	0	0	0	
110 LET D=12550	0	12550	0	0	
120 LET C=4576	0	12550	4576	0	
130 LET B=45885	45885	12550	4576	0	
140 PRINT S	45885	12550	4576	0	0
150 END	45885	12550	4576	0	

Este programa no encuentra el balance final correcto S, porque B, D, C no tienen asignados valores cuando se calcula S.

6.

	B	C	A	TRC
	0	0	0	
100 LET B=24	34	0	0	
110 LET C=110	34	110	0	

	B	C	A	TRC
120 LET A = 180 – B – C	34	110	36	
130 PRINT A	34	110	36	36
140 END	34	110	36	

7.

	B	C	A	TRC
	0	0	0	
100 LET A = 180 – B – C	0	0	180	
110 LET C = 110	0	110	180	
120 LET B = 34	34	110	180	
130 PRINT A	34	110	180	180
140 END	34	110	180	

Cambiando las líneas 100 y 120, se le asigna valor a A mientras B y C contienen cero.

8.

```
20 LET C = 90
30 LET A = 26.72
40 LET B = 180 – A – C
50 PRINT B
60 END
```

9.

	P	D	R	I	TRC
	0	0	0	0	
100 LET P = 6000	6000	0	0	0	
110 LET D = 1000	6000	1000	0	0	
120 LET R = .015	6000	1000	.015	0	
130 LET I = (P – D)*R	6000	1000	.015	75	
140 PRINT I	6000	1000	.015	75	75
150 END	6000	1000	.015	75	

10.

```
100 LET P = 6000
110 LET D = 1000
120 LET R = .015
130 LET I = (P – B)*R
140 PRINT "INTERESES DE LA DEUDA EN EL PRIMER MES"
150 PRINT I
160 LET B = P – D + I – 250
170 PRINT "CANTIDAD TODAVIA DEBIDA"
180 PRINT B
190 END
```

11.

	P	R	I	TRC
	0	0	0	
5 LET P=5000	5000	0	0	
10 LET R=.015	5000	.015	0	
15 LET I=P*R	5000	.015	75	
20 LET P=P+I	5075	.015	75	
25 PRINT "INTERESES OBTENIDOS"	5075	.015	75	INTERESES OBTENIDOS
30 PRINT I	5075	.015	75	75
35 PRINT "CAPITAL DESPUES DE UN MES"	5075	.015	75	CAPITAL DESPUES DE UN MES
40 PRINT P	5075	.015	75	5075
45 END	5075	.015	75	

12. Cambie las líneas 5 y 10 por:

```
 5 LET P=4800
10 LET R=.02
```

13.

```
 5 LET P=4800
 10 LET R=.02
 20 LET I=P*R
 40 LET P=P+I
 50 PRINT "INTERESES OBTENIDOS EL PRIMER MES"
 60 PRINT I
 70 PRINT "EL CAPITAL AL FINAL DEL PRIMER MES ES"
 80 PRINT P
 90 LET I=P*R
100 LET P=P+I
110 PRINT "INTERESES OBTENIDOS EL SEGUNDO MES"
120 PRINT I
130 PRINT "EL CAPITAL AL FINAL DEL SEGUNDO MES ES"
140 PRINT P
150 END
```

```
]RUN
INTERESES GANADOS EL PRIMER MES SON
96
EL CAPITAL AL FINAL DEL PRIMER MES ES
4896
INTERESES GANADOS EL SEGUNDO MES SON
97.92
CAPITAL AL FINAL DEL SEGUNDO MES ES
4993.92
```

14.

	IBM	P1	ITT	P2	T	TRC
	0	0	0	0	0	
100 LET IBM = 120	120	0	0	0	0	
200 LET P1 = 76.75	120	7675	0	0	0	
300 LET ITT = 76	120	7675	76	0	0	
500 LET P2 = 23.50	120	7675	76	23.5	0	
550 LET T = IBM*P1 + ITT*P2	120	7675	76	23.5	10996	
575 PRINT "LA INVERSION TOTAL DEL SEÑOR JONES ES"	120	7675	76	23.5	10996	LA INVERSION TOTAL DEL SEÑOR JONES ES
600 PRINT T	120	7675	76	23.5	10996	10996
800 END	120	7675	76	23.5	10996	

15. Añada las siguientes líneas BASIC al programa:

```
510 LET FM = 250
520 LET P3 = 65.25
550 LET T = IBM*P1 + ITT*P2 + FM*P3
```

Ejercicios 3.5

1.

	A	D	H	M	TRC
	0	0	0	0	
70 LET A = 5	5	0	0	0	
80 LET D = 365*A	5	1825	0	0	
90 LET H = 24*D	5	1825	43800	0	
120 LET M = 60*H	5	1825	43800	2628000	
130 PRINT "EL NUMERO DE MINUTOS ES"	5	1825	43800	2628000	EL NUMERO DE MINUTOS ES
140 PRINT M	5	1825	43800	2628000	2628000
150 END	5	1825	43800	2628000	

2.

```
10 REM M ES EL NUMERO DE MINUTOS
20 REM H ES EL NUMERO DE HORAS
30 REM D ES EL NUMERO DE DIAS
40 REM A ES EL NUMERO DE AÑOS
50 LET M = 2628000
60 LET H = M/60
70 LET D = H/24
```

```
 80 LET A=D/365
 90 PRINT "EL NUMERO DE AÑOS ES"
100 PRINT A
110 END
```

Ejercicios 3.7

1. *a)*
```
 20 REM MODULO DE ASIGNACION
 30 LET X=24
 40 LET Y=-4
 50 REM MODULO DE CALCULO
 60 LET Z=X+Y
 70 LET T=2*X-Y^3
 80 REM MODULO DE SALIDA
 90 PRINT X
 95 PRINT Z
100 PRINT T
110 END
```

b)
```
 10 REM MODULO DE ASIGNACION
 20 LET Q$="MI"
 30 LET R$="NOMBRE"
 40 LET E1$="ES"
 50 LET W$="MOZART"
 60 REM MODULO DE SALIDA
 70 PRINT Q$
 80 PRINT R$
 90 PRINT E1$
100 PRINT W$
110 PRINT "!"
120 END
```

Capítulo 4

Ejercicios 4.2

1. *a)* 25 5 625
255625
25 5 625
55-8324 ES EL TELEFONO DEL SR. TALKER
c) ANTISISTEMARIANISMO TIENE 19 LETRAS

Capítulo 5

Ejercicios 5.1

1. *a)*

	A	D	F1	H	TRC
	0	0	0	0	
200 READ A,D,F1	5	6	-2	0	
210 LET H=2*A-D/2*F1	5	6	-2	16	
220 PRINT H	5	6	-2	16	(16)
230 DATA 5,6,-2	5	6	-2	16	
240 END	5	6	-2	16	

b)

	R$	T	E	F$	TRC
		0	0		
30 READ R$,T,E,F$	TEA	4	2	?	
35 PRINT R$,T,E;	TEA	4	2	?	TEA 4 2
40 PRINT F$	TEA	4	2	?	TEA 4 2?
50 DATA "TEA"	TEA	4	2	?	
60 DATA 4,2	TEA	4	2	?	
70 DATA "?"	TEA	4	2	?	
80 END	TEA	4	2	?	

c)

	X	Y	Z	TRC
10 DATA 2,3,4	0	0	0	
20 READ X,Y,Z	2	3	4	
30 LET Z=X+Y+Z	2	3	9	
40 PRINT Z	2	3	9	9
50 END	2	3	9	

2. *a)* No funciona. A T, que es variable numérica, se le ha asignado el valor "BIEN".
b) No funciona. No hay datos suficientes para todas las sentencias READ.
c) La salida será: MOZART.
d) No funciona. No hay sentencia DATA para la sentencia READ.
e) Salida: 7 7 7
f) Salida: 3 5

3. *a)*
```
100 REM BALANCE DEL TALONARIO DE CHEQUES
110 READ B,D,C
120 LET S=B+D-C
130 PRINT S
140 DATA 458.65, 125.50, 45.76
150 END
```
b)
```
100 REM SOLUCION PARA UN ANGULO
110 READ B,C
120 LET A=180-B-C
130 PRINT A
140 DATA 34,110
150 END
```
c)
```
100 REM CARGO DE INTERESES
110 READ P,D,R
120 LET I=(P-D)*R
130 PRINT I
140 DATA 6000,1000,.015
150 END
```
d)
```
100 REM ANALISIS DE LAS EXISTENCIAS DE MERCADO
110 READ IBM, P1, ITT, P2
120 LET T=IBM*P1+ITT*P2
130 PRINT "LA INVERSION TOTAL DEL SR. RUBIO ES"
```

```
140 PRINT T
150 DATA 120, 76.75, 75, 23.50
160 END
```

Ejercicios 5.2

1.
```
100 REM CALCULA SOLO LA EDAD MEDIA DE LAS HIJAS
110 LET S=0
120 READ N$,A
130 READ N$,A
135 PRINT N$;" TIENE ";A;" AÑOS."
140 LET S=S+A
150 READ N$,A
155 PRINT N$;" TIENE ";A;" AÑOS."
160 LET S=S+A
170 READ N$,A
180 READ N$,A
185 PRINT N$;" TIENE ";A;" AÑOS."
190 LET S=S+A
200 LET M=S/3
210 PRINT "LA EDAD MEDIA DE LAS HIJAS ES ";M
220 DATA "LUIS",6,"CRISTINA",8,"MARIANA",10
230 DATA "IRO",12,"ANA",13,"PABLO",18
240 END
```

```
]RUN
10.3333333
```

Ejercicios 5.3

1.
```
110 REM NOTA MEDIA DE CADA ESTUDIANTE
120 READ A,B,C
130 PRINT "LA NOTA MEDIA DE DANIEL ES "; (A+B+C)/3
140 READ A,B,C
150 PRINT "LA NOTA MEDIA DE BLAS ES "; (A+B+C)/3
160 READ A,B,C
170 PRINT "LA NOTA MEDIA DE JUAN ES "; (A+B+C)/3
180 READ A,B,C
190 PRINT "LA NOTA MEDIA DE JOSE ES "; (A+B+C)/3
200 RESTORE
210 REM CALCULO DE LA NOTA MEDIA DE CADA EXAMEN
220 LET S1=0 :REM ACUMULADOR PARA CONTAR 1 PUNTO
230 LET S2=0 :REM ACUMULADOR PARA CONTAR 2 PUNTOS
240 LET S3=0 :REM ACUMULADOR PARA CONTAR 3 PUNTOS
250 READ A,B,C
260 LET S1=S1+A
270 LET S2=S2+B
280 LET S3=S3+C
290 READ A,B,C
300 LET S1=S1+A
310 LET S2=S2+B
320 LET S3=S3+C
330 READ A,B,C
340 LET S1=S1+A
350 LET S2=S2+B
360 LET S3=S3+C
370 READ A,B,C
```

```
380 LET S1=S1+A
390 LET S2=S2+B
400 LET S3=S3+C
410 PRINT "MEDIA DEL TEST 1=";S1/4
420 PRINT "MEDIA DEL TEST 2=";S2/4
430 PRINT "MEDIA DEL TEST 3=";S3/4
440 REM
450 DATA 78,89,56,56,66,89,81,55,99,66,78,80
460 END
```

Ejercicios 5.4

1.
```
100 REM ANALISIS DE PAQUETES DE ACCIONES
110 REM
115 REM MODULO DE ASIGNACION
120 INPUT "CUANTAS ACCIONES DE IBM? ";S1
130 INPUT "CUANTO POR ACCION? ";D1
140 INPUT "CUANTAS ACCIONES DE SONY? ";S2
150 INPUT "CUANTO POR ACCION? ";D2
160 INPUT "CUANTAS ACCIONES DE ITT? ";S3
170 INPUT "CUANTO POR ACCION? ";D3
180 REM MODULO DE CALCULO
190 LET I1=S1*D1
200 LET I2=S2*D2
210 LET I3=S3*D3
220 LET T=I1+I2+I3
230 REM
240 REM MODULO DE SALIDA
260 PRINT
270 PRINT "SU INVERSION EN IBM ES ";I1
280 PRINT "SU INVERSION EN SONY ES ";I2
290 PRINT "SU INVERSION EN ITT ES ";I3
300 PRINT "SU INVERSION TOTAL ES ";T
310 END
```

2.
```
  5 INPUT "CUAL ES EL CAPITAL INVERTIDO? ";P
 10 INPUT "CUAL ES EL INTERES MENSUAL? ";R
 20 LET I=P*R
 30 LET P=P+I
 40 PRINT "INTERESES OBTENIDOS: ";I
 50 PRINT "CANTIDAD EN CUENTA DESPUES DE UN MES: ";P
 60 END
```

Capítulo 6

Ejercicios 6.1

1. *a*) Falso; *b*) Verdadero *c*) Verdadero; *d*) Falso
2. *a*) El valor numérico 2 no puede ser comparado con la variable tipo cadena X$.
   b) La cadena "4" no puede ser comparada con la variable numérica B.
   c) PRINT R no es una condición.
   d) Una condición no puede ir detrás de THEN.

3.

	X	X1$	T	Z$	TRC
	0		0		
100 LET X = 2	2		0		
110 LET X1$ = "HOLA"	2	HOLA	0		
120 LET T = 4	2	HOLA	4		
130 LET Z$ = "ADIOS"	2	HOLA	4	ADIOS	
140 IF X + 1 < > 2 THEN LET T = 0	2	HOLA	0	ADIOS	
150 IF T = 0 THEN PRINT X	2	HOLA	0	ADIOS	2
160 IF X1$ < Z$ THEN LET Z$ = "MAÑANA"	2	HOLA	0	MAÑANA	
170 IF T > X + 6 THEN LET T = 5	2	HOLA	0	MAÑANA	
180 END	2	HOLA	0	MAÑANA	

4. *a)*

	R	V1	K$
	0	0	
100 LET R = 2	2	0	
110 LET V1 = 45	2	45	
120 LET K$ = "EL"	2	45	EL
130 IF K$ > "ELLA" OR V1 = 100 THEN LET R = R + 1	2	45	EL
140 IF K$ < "ELLA" OR V1 = 100 THEN LET R = R − 2	0	45	EL
150 IF K$ < "ELLA" AND V1 < > 100 THEN LET R = R − 2	−2	45	EL
160 END	−2	45	EL

b)

	G3$	T$	R	Y
			0	0
10 LET G3$ = "GATO"	GATO		0	0
20 LET T$ = "PERRO"	GATO	PERRO	0	0
30 LET R = 3	GATO	PERRO	3	0
40 IF G3$ < > "KITTY" AND R < 2 THEN LET Y = Y + R * 2	GATO	PERRO	3	0

	G3$	T$	R	Y
50 IF G3$ < > "KITTY" OR R < 2 THEN LET Y = Y + R*2	GATO	PERRO	3	6
60 IF (R = 5 AND T$ = "PERRO") OR T$ < > "MESTIZO" THEN LET T$ = "CHINO"	GATO	MESTIZO	3	6
70 IF (R = 5 OR T$ = "CACHORRO") AND T$ < > "MESTIZO" THEN LET T$ = "HUESO"	GATO	MESTIZO	3	6
80 END	GATO	MESTIZO	3	6

5. *a)*

	X	Y	L	TRC
	0	0	0	
10 INPUT X,Y	35	72	0	? 35,72
20 LET L = X	35	72	35	
30 IF Y > L THEN LET L = Y	35	72	72	
40 PRINT "EL MAYOR ES ";L	35	72	72	EL MAYOR ES 72
50 END	35	72	72	

b) Cambie a las siguientes líneas:

```
30 IF Y<L THEN LET L=Y
40 PRINT "EL MENOR ES" ;L
```

c)

```
10 INPUT X,Y
20 LET L=X
30 IF Y>L THEN LET L=Y
40 IF X<>Y THEN PRINT "EL MAYOR ES ";L
50 IF X=Y THEN PRINT "SON IGUALES"
60 END
```

d)

```
10 INPUT X,Y
20 IF X>Y THEN PRINT "EL MAYOR ES ";X
30 IF X<Y THEN PRINT "EL MAYOR ES ";Y
40 END
```

6. *a)*

	X	Y	Z	L	TRC
	0	0	0	0	
120 INPUT X,Y,Z	5	23	17	0	? 5,23,17
130 LET L = X	5	23	17	5	
140 IF L < Y THEN LET L = Y	5	23	17	23	
150 IF L < Z THEN LET L = Z	5	23	17	23	
160 PRINT "EL MAYOR ES" ;L	5	23	17	23	EL MAYOR ES 23
170 END	5	23	17	23	

b) La variable L siempre toma el mayor valor. Si se encuentra un número mayor, éste se sitúa en L.

c)
```
100 REM ENCUENTRA EL MAS PEQUEÑO DE 4 NUMEROS
120 INPUT X,Y,Z,W
130 LET L=X
140 IF L>Y THEN LET L=Y
150 IF L>Z THEN LET L=Z
160 IF L>W THEN LET L=W
170 PRINT "EL MAS PEQUEÑO ES ";L
180 END
```

7. *a*) Si se escribe algo distinto de CACAHUETES, NUECES o ANACARDOS el valor de X se mantiene en cero. La línea 130 detecta esto, escribe un mensaje y para el programa. Los buenos programas deben comprobar las entradas inesperadas.

b) Borre las líneas 110 a 200 e incluya estas líneas:

```
110 INPUT "CUANTOS KILOS QUIERE DE CACAHUETES? ";P
120 INPUT "CUANTOS KILOS QUIERE DE NUECES? ";W
130 INPUT "CUANTOS KILOS QUIERE DE ANACARDOS? ";C
140 LET Q=185*P+234*W+485*C
150 PRINT "EL PRECIO TOTAL ES ";Q
160 END
```

c) Haga estos cambios en el programa original:

```
155 READ A1,A2,A3
160 IF N$="CACAHUETES" THEN LET Q=P*A1
170 IF N$="NUECES" THEN LET Q=P*A2
180 IF N$="ANACARDOS" THEN LET Q=P*A3
200 DATA 185,234,485
210 END
```

d) De tal manera que la entrada esté en la misma línea que la pregunta.

e) Cambie y añada estas líneas:

```
120 IF N$="CACAHUETES" THEN LET X=1
122 IF N$="NUECES" THEN LET X=1
125 IF N$="ANACARDOS" THEN LET X=1
```

Ejercicios 6.2

1. No hay diferencia, se pulse SI o NO (ver número 2).

2. Fuerza al usuario a pulsar SI para acabar el programa. Cualquier otra cosa hará empezar el programa de nuevo. En un buen programa, las decisiones drásticas, como acabar el programa, requerirían respuestas específicas. Otras respuestas deben causar el menor daño, esto es, volver a empezar el programa. Si la línea 50 se sustituye por 50 IF D$="NO" THEN GOTO 20, entonces entradas inesperadas pueden acabar el programa, lo cual no es deseable.

3.
```
110 REM MODULO DE ASIGNACION
115 LET S=0
```

```
120 LET A=3
125 REM MODULO DE CALCULO
130 LET S=S+A
140 IF A=1001 THEN GOTO 180
160 LET A=A+2
170 GOTO 130
175 REM MODULO DE SALIDA
180 PRINT "LA SUMA ES ";S
190 END
```

4.
```
140 REM MODULO DE ASIGNACION
150 LET S=0
160 LET D=1
170 REM MODULO DE CALCULO
190 LET S=S+(1/D)^2
200 LET D=D+1
210 IF D<51 THEN GOTO 190
220 REM MODULO DE SALIDA
230 PRINT "LA SUMA ES ";S
240 END
```

5. Un error del tipo OUT OF DATA podría ocurrir si sólo hubiera un asterisco al final de la lista de datos. El segundo asterisco puede ser una cadena. Se introduce en Y$, pero no se usa.

6. Suponga que son utilizadas las mismas sentencias DATA (líneas 20-45).

```
 50 INPUT "ESCRIBA EL NUMERO DE TELEFONO ";N$
 60 READ X$,Y$
 70 IF Y$="*" THEN PRINT "TAL TELEFONO NO EXISTE": GOTO 100
 80 IF Y$<>N$ THEN GOTO 60
 90 PRINT "PARA EL TELEFONO ";N$;" EL NOMBRE ES ";X$
100 END
```

7. Cambie estas líneas:

```
 20 DATA "TIENDA DE MANTEQUERIA DE JOSE", "S. GINES, 4", "234-5678" (etc.)
 40 DATA "BANCO CANTABRICO", "VERA, 145","445-5678"
 45 DATA "*","*","*"
 60 READ X$,Y$,Z$
100 PRINT N$" EL NUMERO ES ";Z$
105 PRINT "LA DIRECCION ES ";Y$
```

8. Aunque no se haga depósito, la variable D es puesta a cero. Por tanto, la línea 50 funciona de todas formas.

9.
```
110 INPUT "INTRODUZCA EL SALDO ACTUAL ";B
120 INPUT "DESEA HACER UN DEPOSITO? ";A$
125 LET D=0
130 IF A$="SI" THEN INPUT "HAGA EL DEPOSITO: ";D
135 LET B=B+D
140 INPUT "INTRODUZCA EL VALOR DEL CHEQUE (HAGA O CUANDO TERMINE) ";C
150 IF C=0 THEN GOTO 190
160 LET B=B-C
165 GOTO 140
190 INPUT "DESEA HACER OTRA ENTRADA? ";A$
200 IF A$="SI" THEN GOTO 125
210 PRINT "SU NUEVO SALDO ES ";B
220 END
```

Ejercicios 7.1

1.

	Q	R$	TRC
110 FOR Q=1 TO 3	1		
PRIMER LAZO			
120 READ R$	1	CHINA	
130 PRINT R$	1	CHINA	CHINA
140 NEXT Q	2	CHINA	
SEGUNDO LAZO			
120 READ R$	2	2	
130 PRINT R$	2	2	2
140 NEXT Q	3	2	
TERCER LAZO			
120 READ R$	3	EUROPA	
130 PRINT R$	3	EUROPA	EUROPA
140 NEXT Q	4	EUROPA	
150 END	4	EUROPA	

2.

	I	K	TRC
20 FOR I=1 TO 10	1	0	
PRIMER LAZO			
30 LET K=2*I	1	2	
40 IF K>4 THEN GOTO 60	1	2	
50 NEXT I	2	2	
SEGUNDO LAZO			
30 LET K=2*I	2	4	
40 IF K>4 THEN GOTO 60	2	4	
50 NEXT I	3	4	
TERCER LAZO			
30 LET K=2*I	3	6	

	I	K	TRC
40 IF K>4 THEN GOTO 60	3	6	
LAZO FINAL			
60 PRINT "ESO ES TODO!"	3	6	ESO ES TODO!
70 END	3	6	

3.
```
100 FOR I=3 TO 6
110 LET P=2*I-5
115 PRINT P
120 NEXT I
130 END
```

4. *a*) 1.2, 1.4, 1.6, 1.8, 2, 2.2, 2.4, 2.6
b) 3.5, 3, 2.5, 2, 1.5, 1, .5, 0
c) 3, 6
d) 8, 12.25, 9, 6.25, 4, 2.25, .25, 0

Ejercicios 7.2

1.
```
20 REM MODULO DE ASIGNACION
30 INPUT "CUAL ES EL CAPITAL INICIAL?";P
40 INPUT "CUAL ES EL INTERES?";R
50 INPUT "POR CUANTOS AÑOS?";N
55 PRINT
60 REM MODULOS DE CALCULO Y SALIDA
65 REM LA SIGUIENTE LINEA CONVIERTE UN TIPO DE INTERES TAL
68 REM COME 15 POR CIENTO A .15
70 IF R>1 THEN LET R=R/100
80 FOR I=1 TO N
90 LET P=P+P*R
100 PRINT "EL CAPITAL TOTAL DESPUES DE ";I;" AÑOS ES ";P
110 NEXT I
120 END
```

```
]RUN
CUAL ES EL CAPITAL INICIAL? 1000
CUAL ES EL TIPO DE INTERES? 6.5
DURANTE CUANTOS AÑOS? 10

TOTAL DESPUES DE 1 AÑOS ES  1065
TOTAL DESPUES DE 2 AÑOS ES  1134.225
TOTAL DESPUES DE 3 AÑOS ES  1207.94963
TOTAL DESPUES DE 4 AÑOS ES  1286.46635
TOTAL DESPUES DE 5 AÑOS ES  1370.08666
TOTAL DESPUES DE 6 AÑOS ES  1459.1423
TOTAL DESPUES DE 7 AÑOS ES  1553.98655
TOTAL DESPUES DE 8 AÑOS ES  1654.99567
TOTAL DESPUES DE 9 AÑOS ES  1762.57039
TOTAL DESPUES DE 10 AÑOS ES  1877.13747
```

2. Añada las siguientes instrucciones:

```
 95 LET V=P*R+V
115 PRINT "EL INTERES TOTAL OBTENIDO ES ";V
```

3.
```
10 INPUT "CUANTOS TERMINOS? ";M
20 LET S=0
30 FOR K=1 TO M
40 LET S=S+ 1/(K*(K+1)*(K+2))
50 NEXT K
60 PRINT "LA SUMA DE ";M;" TERMINOS ES ";S
70 END
```

```
]RUN
CUANTOS TERMINOS? 10
LA SUMA DE 10 TERMINOS ES .246212121
```

```
]RUN
CUANTOS TERMINOS? 100
LA SUMA DE 100 TERMINOS ES .249951466
```

```
]RUN
CUANTOS TERMINOS? 1000
LA SUMA DE 1000 TERMINOS ES .249999501
```

4.
```
100 REM CALCULAR LA SUMA DE LOS NUMEROS DE UNA LISTA DE DATA
105 LET S=0
110 FOR I=1 TO 10
120 READ X
130 LET S=S+X
140 NEXT I
150 REM CALCULO DE LA MEDIA
160 LET A=S/10
170 RESTORE
180 REM CALCULA E IMPRIME LOS NUMEROS INFERIORES A LA MEDIA
190 FOR I=1 TO 10
200 READ X
210 IF A>=X THEN PRINT X
220 NEXT I
230 DATA 567,45,668,3322,-234,44,6789,-34,67,77
240 END
```

```
]RUN
567
85
668
-234
44
-34
67
77
```

5.
```
100 LET S=0
110 FOR I=1 TO 50
120 LET X=INT(RND(1)*100)+1
130 LET S=S+X
```

```
140 NEXT I
145 LET S=S/50
150 PRINT "LA MEDIA DE CINCUENTA NUMEROS ALEATORIOS ES ";S
160 END
```

6. Usando las mismas sentencias DATA puede escribirse

```
(110-140 como antes)
150 INPUT "CUAL ES EL NUMERO DE TELEFONO? ";T$
160 FOR I=1 TO 5
170 READ X$,Y$
180 IF Y$<>T$ THEN GOTO 215
190 PRINT "EL NOMBRE ES ";X$
200 LET E=1
210 LET I=5
215 NEXT I
220 IF E=1 THEN GOTO 240
230 PRINT "LO SIENTO, NO HAY TAL TELEFONO"
240 END
```

```
]RUN
CUAL ES EL NUMERO DE TELEFONO? 235-6666
EL NOMBRE ES MARI JUANA
```

7. Cambie y añada las siguientes líneas:

```
102 REM LA SIGUIENTE SENTENCIA DATA ES EL
105 REM NUMERO DE NOMBRES EN EL DIRECTORIO
107 DATA 5
145 READ N
160 FOR I=1 TO N
```

Ejercicios 7.3

1. La sentencia PRINT vacía invalida el punto y y coma, así que la siguiente línea de números aparece en otra línea.
2.

```
100 FOR I=1 TO 4
110 FOR J=1 TO 5
120 PRINT I;"+";J;"=";I+J;"      ";
140 NEXT J
145 PRINT
150 NEXT I
160 END
```

Capítulo 8

Ejercicios 8.1

1.

	I	C(1)	C(2)	C(3)	S	TRC
	0	0	0	0	0	
110 FOR I=1 TO 3	1	0	0	0	0	
120 INPUT "PULSA UN NUMERO: ";C(I)	1	23	0	0	0	PULSA UN NUMERO: 23
130 NEXT I	2	23	0	0	0	

	I	C(1)	C(2)	C(3)	S	TRC
120 INPUT "PULSA UN NUMERO ";C(I)	2	23	5	0	0	PULSA UN NUMERO: 5
130 NEXT I	3	23	5	0	0	
120 INPUT "PULSA UN NUMERO: ";C(I)	3	23	5	−22	0	PULSA UN NUMERO: −22
130 NEXT I	4	23	5	−22	0	
140 LET S=0	4	23	5	−22	0	
150 FOR I=1 TO 3	1	23	5	−22	0	
160 LET S=S+C(I)	1	23	5	−22	23	
170 NEXT I	2	23	5	−22	23	
160 LET S=S+C(I)	2	23	5	−22	28	
170 NEXT I	3	23	5	−22	28	
160 LET S=S+C(I)	3	23	5	−22	6	
170 NEXT I	4	23	5	−22	6	
190 PRINT "SUMA= ";S	4	23	5	−22	6	SUMA=6

200 END

2. He aquí una de las posibles soluciones:

LET A(1)=(A(4)−A(2))/A(2)
LET A(2)=A(1)+A(1)
LET A(4)=A(2)+A(2)+A(4)
LET A(3)=A(4)+A(2)
LET A(2)=A(2)−A(2)

3.

	K	I	G(1)	G(2)	G(3)
10 DATA 3,2	0	0	0	0	0
15 READ I,K	2	3	0	0	0
20 LET G(I)=10	2	3	0	0	10
25 LET G(K)=4	2	3	0	4	10
30 LET G(I−K)=5*G(I)	2	3	50	4	10
35 RESTORE	2	3	50	4	10
40 READ G(K),K	2	3	50	3	10
45 LET G(K−1)=G(G(K))	2	3	10	3	10

4. 90 MODULO DE ASIGNACION
100 FOR K=1 TO 10
110 INPUT "ESCRIBE UN NUMERO ";B(K)

```
120 NEXT K
125 REM MODULO DE CALCULO
130 LET S=0
140 FOR K=1 TO 10
150 LET S=S+B(K)
160 NEXT K
170 LET A=S/10
175 REM MODULO DE SALIDA
180 PRINT "LOS SIGUIENTES NUMEROS SON MAYORES QUE ";A
185 FOR I=1 TO 10
190 IF A<B(I) THEN PRINT B(I)
200 NEXT I
205 PRINT
210 PRINT "LOS SIGUIENTES NUMEROS SON MENORES O IGUALES QUE ";A
220 FOR I=1 TO 10
230 IF A>=B(I) THEN PRINT B(I)
240 NEXT I
250 END
```

Ejercicios 8.2

1.
```
100 DIM A(30),E1(30),E2(30),E3(30)
110 REM ALMACENA CALIFICACIONES Y MEDIA DE CADA ALUMNO
120 FOR I=1 TO 30
130 PRINT "ESTUDIANTE NUMERO ";I
140 INPUT "CUALES SON LAS CALIFICACIONES? ";E1(I),E2(I),E3(I)
145 LET A(I)=(E1(I)+E2(I)+E3(I))/3
150 PRINT
160 NEXT I
170 REM ESCRIBE LOS NUMEROS DE LOS ESTUDIANTES CUYA
180 REM MEDIA ES SUPERIOR A 80
190 PRINT "LOS SIGUIENTES ESTUDIANTES TIENEN LA
    MEDIA SUPERIOR A 80"
200 FOR I=1 TO 30
210 IF A(I)>80 THEN PRINT I
220 NEXT I
230 END
```

2. Sin el paréntesis el ordenador asociaría la letra A con la T de THEN, y lo considera como la palabra reservada AT, lo cual produce un SYNTAX ERROR.

3.
```
100 REM ALMACENA N NUMEROS ALEATORIOS ENTRE 1 Y 100
100 INPUT "CUANTOS NUMEROS ALEATORIOS DEBEN SER CALCULADOS? ";N
120 DIM R(N)
125 REM MODULO DE ASIGNACION
130 FOR I=1 TO N
140 LET R(I)=INT (RND(1)*100)+1
150 NEXT I
160 REM CALCULA LA MEDIA DE LOS N NUMEROS
165 LET S=0
170 REM MODULO DE CALCULO
180 FOR I=1 TO N
190 LET S=S+R(I)
200 NEXT I
210 LET A=S/N
220 REM MODULO DE SALIDA
230 REM ESCRIBE LOS NUMEROS MAYORES QUE LA MEDIA
```

```
235 PRINT "LOS SIGUIENTES NUMEROS SON MAYORES QUE ";A
240 FOR I=1 TO N
250 IF A<R(I) THEN PRINT R(I)
260 NEXT I
270 REM ESCRIBE LOS NUMEROS PARES
280 PRINT "LOS SIGUIENTES SON LOS NUMEROS PARES"
290 FOR I=1 TO N
300 IF INT(R(I)/2)=R(I)/2 THEN PRINT R(I)
310 NEXT I
320 END
```

4.
```
100 REM ALMACENAMIENTO Y PROMEDIO DE N NUMEROS ALEATORIOS
105 INPUT "DE CUANTOS NUMEROS HAY QUE HACER MEDIA? ";N
110 DIM W(N)
120 LET S=0
130 LET C=0
140 FOR K=1 TO N
150 LET W(K)=RND(1)
160 S=S+W(K)
170 NEXT K
180 LET A=S/N
190 PRINT "LA MEDIA ES ";A
200 FOR K=1 TO N
210 IF (W(K)>A) THEN PRINT W(K): LET C=C+1
220 NEXT K
230 PRINT C;" NUMEROS SON MAYORES QUE LA MEDIA."
240 END
```

5.
```
100 INPUT "CUANTOS NUMEROS VAN A USARSE PARA EL CALCULO DE LA
    DESVIACION TIPICA? ";N
105 DIM A(N),B(N)
110 REM ENTRADA DE N NUMEROS EN LA MATRIZ A
120 FOR I=1 TO N
130 INPUT A(I)
140 NEXT I
150 REM REALIZA PASO (A)
155 LET S=0
160 FOR I=1 TO N
170 LET S=S+A(I)
180 NEXT I
190 REM USAMOS A PARA LA MEDIA. LA VARIABLE A
200 REM ES DISTINTA QUE LA MATRIZ A
210 LET A=S/N
220 REM REALIZA PASO (B)
230 FOR J=1 TO N
240 LET B(J)=A(J)-A
250 NEXT J
260 REM REALIZA PASO (C)
270 FOR J=1 TO N
280 LET B(J)=B(J)^2
290 NEXT J
300 REM REALIZA PASO (D)
310 LET S=0
320 FOR I=1 TO N
330 LET S=S+B(I)
340 NEXT I
```

```
350 LET S=S/N
360 REM REALIZA PASO (E)
370 LET S=SQR(S)
380 REM SALIDA DE LA DESVIACION TIPICA
390 PRINT S
400 END
```

Ejercicios 8.3

1. Hacen falta cinco pasadas para ordenar seis números. Hacen falta N − 1 pasadas para ordenar N números.

2.
```
100 REM ORDENACION DESCENDENTE
115 DIM A(10)
120 DATA 23,-56,100,45,56,34,444,67,0,45
130 FOR I=1 TO 10
140 READ A(I)
150 NEXT I
160 FOR K=1 TO 9
170 FOR J=1 TO 9
190 IF A(J)<A(J+1) THEN LET L=A(J) :LET A(J)=A(J+1) :
    LET A(J+1)=L
230 NEXT J
240 NEXT K
250 REM HECHA LA ORDENACION, ESCRIBIMOS MATRIZ
260 FOR I=1 TO 10
270 PRINT A(I);"  ";
280 NEXT I
300 END
```

3. El programa sólo realizará una pasada sobre los números, haciendo que el mayor número de la matriz A sea llevado a la posición A(4).

4.
```
100 REM ORDENACION ASCENDENTE
110 REM DE UNA CANTIDAD ARBITRARIA DE NUMEROS
120 INPUT "CUANTOS NUMEROS VA A INTRODUCIR? ";N
125 DIM A(N)
130 FOR I=1 TO N
140 INPUT "ESCRIBA UN NUMERO: ";A(I)
150 NEXT I
160 REM ORDENACION DE LOS NUMEROS
170 FOR K=1 TO N-1
180 FOR J=1 TO N-1
190 IF A(J)>A(J+1) THEN LET L=A(J) :LET A(J)=A(J+1) :
    LET A(J+1)=L
200 NEXT J
210 NEXT K
220 REM HECHA LA ORDENACION, ESCRIBIMOS MATRIZ
230 FOR I=1 TO N
240 PRINT A(I);" ";
250 NEXT I
260 END
```

5. En el programa del problema 4, cambie la línea 180 por:

```
180 FOR J=1 TO N-K
```

Ejercicios 8.4

1.

	N	I	R(1)	R(2)	R(3)	K	TRC
125 LET I=0	0	0	0	0	0	0	
130 INPUT "ESCRIBE UN NUMERO DECIMAL ";N	6	0	0	0	0	0	ESCRIBE UN NUMERO DECIMAL: 6
140 LET I=I+1	6	1	0	0	0	0	
150 LET R(I)=N−2* INT(N/2)	6	1	0	0	0	0	
160 LET N=INT(N/2)	3	1	0	0	0	0	
140 LET I=I+1	3	2	0	0	0	0	
150 LET R(I)=N−2* INT(N/2)	3	2	0	1	0	0	
160 LET N=INT(N/2)	1	2	0	1	0	0	
140 LET I=I+1	1	3	0	1	0	0	
150 LET R(I)=N−2* INT(N/2)	1	3	0	1	1	0	
160 LET N=INT(N/2)	0	3	0	1	1	0	
190 FOR K=I TO 1 STEP−1	0	3	0	1	1	3	
200 PRINT R(K);	0	3	0	1	1	3	1
210 NEXT K	0	3	0	1	1	2	
200 PRINT R(K);	0	3	0	1	1	2	11
210 NEXT K	0	3	0	1	1	3	
200 PRINT R(K);	0	3	0	1	1	3	110
210 NEXT K	0	3	0	1	1	4	
220 END	0	3	0	1	1	4	

2.
```
100 REM CONVIERTE N DE DECIMAL A BASE OCHO
115 DIM R(100)
120 REM MODULO DE ASIGNACION
125 LET I=0
130 INPUT "ESCRIBA UN NUMERO EN DECIMAL. ";N
140 LET I=I+1
145 REM MODULO DE CALCULO
```

```
150 LET R(I)=N-8*INT(N/8)
160 LET N=INT(N/8)
170 IF N>0 THEN GOTO 140
180 REM MODULO DE SALIDA
190 FOR K=I TO 1 STEP -1
200 PRINT R(K);
210 NEXT K
220 END
```

3.
```
100 REM CONVIERTE N DE DECIMAL A BASE 5 O 7
115 DIM R(100)
120 REM MODULO DE ASIGNACION
125 LET I=0
130 INPUT "ESCRIBA UN NUMERO DECIMAL.";N
135 PRINT "QUIERE QUE EL NUMERO ";N;" SE CONVIERTA A
    BASE 5 O 7?"
138 INPUT B
140 LET I=I+1
145 REM MODULO DE CALCULO
150 LET R(I)=N-B*INT(N/B)
160 LET N=INT(N/B)
170 IF N>0 THEN GOTO 140
180 REM MODULO DE SALIDA
190 FOR K=I TO STEP -1
200 PRINT R(K);
210 NEXT K
220 END
```

4. Cambie la línea 160 por lo siguiente:

```
160 IF (S(I)<=6.5) OR (Y(I)>70) THEN GOTO 170
```

5.
```
100 DATA 6, 67, 6.5, 72, 5.75, 80, 9, 55
110 DATA 8.5, 62, 5, 81, 6, 70, 9, 50
120 DIM S(8),Y(8)
130 FOR K=1 TO 8
140 READ S(K), Y(K)
150 NEXT K
155 LET A=0 :LET C=0
160 REM SCAN ARRAY Y
170 FOR I=1 TO 8
180 IF Y(I)>65 THEN PRINT S(I),Y(I):LET A=A+S(I):LET C=C+1
190 NEXT I
200 PRINT "EL TIEMPO MEDIO DE SUEÑO DE LOS QUE VIVEN
    MAS DE 65 AÑOS ES ";A/C
210 END
```

6.
```
 10 DIM R(50)
 20 FOR I=1 TO 50
 30 LET R(I)=INT(RND(1)*20)+1
 40 NEXT I
 50 LET A=0 :REM A ALMACENA EL NUMERO DE CINCOS
 55 LET B=0 :REM B ALMACENA EL NUMERO DE DIECES
 60 FOR I=1 TO 50
 70 IF R(I)=5 THEN LET A=A+1
 80 IF R(I)=10 THEN LET B=B+1
```

```
 90 NEXT I
100 PRINT "EL NUMERO TOTAL DE CINCOS ES ";A
110 PRINT "EL NUMERO TOTAL DE DIECES ES ";B
120 END
```

7.
```
 10 DATA 34, 67, 345, 35, 678
 20 DIM A(5)
 30 FOR I=1 TO 5
 40 READ A(I)
 50 NEXT I
 60 FOR K=1 TO 30
 65 PRINT "NUMERO DE EJEMPLO ";K
 70 FOR I=1 TO 3
 80 LET R=INT(5*RND(1))+1
 90 PRINT A(R);"  ";
100 NEXT I
110 PRINT
120 NEXT K
130 END
```

8.
```
110 REM ELECCION DE 30 NUMEROS ALEATORIOS ENTRE 1 Y 365
115 REM SEÑALAR SI ALGUNO HA SIDO ELEGIDO DOS VECES
120 DIM A(365)
125 REM INICIALIZAR LA MATRIZ, POR SEGURIDAD
130 FOR J=1 TO 365
140 LET A(J)=0
150 NEXT J
160 REM ELECCION DE LOS NUMEROS ALEATORIOS
180 FOR K=1 TO 30
190 LET R=INT(365*RND(1))+1
200 LET A(R)=A(R)+1
210 NEXT K
220 REM COMPROBAR SI UN NUMERO HA SIDO ELEGIDO DOS VECES
223 REM FIJAR INDICADOR DE FALLO
225 LET F=0
230 FOR J=1 TO 365
240 IF A(J)<=1 THEN GOTO 270
245 REM SOLO SE IMPRIME EXITO SI A(J)>1
250 PRINT "EXITO"
251 REM FIJAR INDICADOR DE EXITO
252 LET F=1
255 REM FIJAR J AL LIMITE SUPERIOR PARA ACABAR BUCLE
260 LET J=365
270 NEXT J
280 IF F=0 THEN PRINT "FRACASO"
290 END
```

9.
```
100 REM COMPROBAR EL PROBLEMA DEL CUMPLEAÑOS
110 REM HACER 100 INTENTOS DE PROGRAMA EN EL PROBLEMA 9
115 DIM A(365)
120 LET S=0
122 REM EMPEZAR 100 PASADAS
123 FOR P=1 TO 100
125 REM INICIALIZAR MATRIZ... AHORA ES NECESARIO
130 FOR J=1 TO 365
140 LET A(J)=0
```

```
150 NEXT J
160 REM ELEGIR NUMEROS ALEATORIOS
170 REM EJEMPLO: SI SE ELIGE EL 253, SUMAR 1 AL ELEMENTO 253
180 FOR K=1 TO 30
190 LET R=INT(365*RND(1))+1
200 LET A(R)=A(R)+1
210 NEXT K
220 REM BUSCAR SI SE HA ELEGIDO UN NUMERO MAS DE UNA VEZ
223 REM FIJAR EL INDICADOR A ERROR
225 LET F=0
230 FOR J=1 TO 365
240 IF A(J)<=1 THEN GOTO 270
245 REM ESCRIBIR EXITO SOLO SI A(J)>1
250 PRINT "EXITO"
251 REM FIJAR EL INDICADOR A EXITO
252 LET F=1
253 REM CONTAR EL NUMERO DE EXITOS
254 LET S=S+1
255 REM FIJAR J AL LIMITE SUPERIOR PARA ACABAR BUCLE
260 LET J=365
270 NEXT J
280 IF F=0 THEN PRINT "FRACASO"
282 REM VOLVER AL PRINCIPIO PARA UNA NUEVA PASADA
283 NEXT P
285 PRINT "HA HABIDO ";S;" EXITOS EN 100 INTENTOS"
290 END
```

10. Esto es una simulación del juego Keno.

```
100 DIM N(80)
110 FOR I=1 TO 80
120 LET N(I)=I
130 NEXT I
140 FOR I= 1 TO 20
150 LET R=INT(80*RND(1))+1
160 IF N(R)=0 THEN GOTO 150
170 PRINT N(R)
180 LET N(R)
190 NEXT I
200 END
```

Ejercicios 8.5

1. La matriz bidimensional P tendrá esta forma:

	columna 1	columna 2
fila 1	45	56
fila 2	2	5

2.

```
210 PRINT "CALCULAR EL COSTE A LO LARGO DE UN AÑO DE:"
220 PRINT "1. LA LUZ"
230 PRINT "2. EL GAS"
240 PRINT "3. EL TELEFONO"
250 INPUT "ESCRIBA EL NUMERO DESEADO";N
260 FOR I=1 TO 12
270 LET S=S+U(N,I)
```

```
280 NEXT I
290 END
```

3.
```
300 INPUT "LISTE TODOS LOS RECIBOS DEL GAS QUE EXCEDAN
    DE QUE CANTIDAD? ";X
310 FOR I=1 TO 12
320 IF X<U(2,I) THEN PRINT U(2,I); "EN EL MES ";I
330 NEXT I
340 END
```

4.
```
300 PRINT "1. RECIBO DE LA LUZ"
310 PRINT "2. RECIBO DEL GAS"
320 PRINT "3. RECIBO DEL TELEFONO"
330 INPUT "ESCRIBA EL NUMERO DESEADO: ";N
340 IF N=1 THEN LET Q$="RECIBOS ELECTRICOS"
350 IF N=2 THEN LET Q$="RECIBOS DEL GAS"
360 IF N=3 THEN LET Q$="RECIBOS DEL TELEFONO"
370 PRINT "LISTOS TODOS LOS ";Q$!" QUE EXCEDAN DE QUE CANTIDAD?";
380 INPUT X
390 FOR I=1 TO 12
400 IF X<U(N,I) THEN PRINT U(N,I);" EN EL MES ";I
410 NEXT I
420 END
```

5. Añada esto al programa:

```
190 REM CALCULO DE LA NOTA MEDIA DE CADA ESTUDIANTE
200 FOR I=1 TO N
210 LET S=0
220 FOR J=1 TO N
230 LET S=S+A(I,J)
240 NEXT J
250 PRINT "LA NOTA MEDIA DEL ESTUDIANTE ";I;" ES ";S/M
260 NEXT I
270 REM CALCULO DE LA NOTA MEDIA DE LA CLASE EN CADA PROBLEMA
280 FOR J=1 TO M
290 LET S=0
300 FOR I=1 TO N
310 LET S=S+A(I,J)
320 NEXT I
330 PRINT "LA NOTA MEDIA DEL PROGRAMA ";J;" ES ";S/N
340 NEXT J
350 END
```

Ejercicios 8.6

1.
```
100 DATA "DALLAS", "CHICAGO", "MIAMI", "BRONX"
105 DIM C$(4)
110 FOR I=1 TO 4
120 READ C$(I)
130 NEXT I
150 REM ORDENACION
160 FOR I=1 TO 3
170 FOR J=1 TO 3
180 IF C$(J)>C$(J+1) THEN LET L$=C$(J) :LET C$(J)=C$(J+1) :
    LET C$(J+1)=L$
190 NEXT J
```

```
200 NEXT I
210 FOR I=1 TO 4
220 PRINT C$(I)
230 NEXT I
240 END
```

2.
```
100 REM ESTE PROGRAMA ACEPTA CADENAS ESCRITAS POR EL USUARIO
110 REM Y LAS ORDENA
120 INPUT "CUANTAS CADENAS VA A INTRODUCIR?";N
130 DIM A$(N)
140 FOR I=1 TO N
150 INPUT "ESCRIBA UNA CADENA: ";A$(I)
160 NEXT I
170 REM ORDENACION DE LA MATRIZ
180 FOR I=1 TO N-1
190 FOR J=1 TO N-1
200 IF A$(J)>A$(J+1) THEN LET L$=A$(J) :LET A$(J)=A$(J+1) :
    LET A$(J+1)=L$
210 NEXT J
220 NEXT I
230 REM IMPRESION DE LA MATRIZ ORDENADA
240 FOR I=1 TO N
250 PRINT A$(I)
260 NEXT I
270 END
```

3.
```
200 INPUT "ENTRE EL NOMBRE (LAS PRIMERAS LETRAS SERVIRAN). ";N$
220 IF N$=LEFT$(K$(I,1),LEN(N$)) THEN PRINT K$(I,1),K$(I,2)
```

4.
```
100 REM ELECCION OBLIGATORIA DE 11 JUGADORES DENTRO
    DE UN EQUIPO DE 14
120 DATA "BRUNO", "JUAN JOSE", "FRANCIS", "JAVIER", "JULIO"
125 DATA "ALBERTO", "HUGO", "ROBERTO", "TONY", "MANOLO"
130 DATA "SANTI", "ABEL", "NACHO", "ENRIQUE"
150 DIM P$(14)
160 FOR J=1 TO 14
170 READ P$(J)
180 NEXT J
185 FOR I=1 TO 11
190 LET L=INT(14*RND(1))+1
210 IF P$(L)="ELEGIDO" THEN GOTO 190
220 PRINT P$(L)
230 LET P$(L)="ELEGIDO"
240 NEXT I
250 END
```

5. Añadir al programa:

```
260 REM CALCULO TOTAL DE LOS DEPOSITOS DE LOS CLIENTES
270 LET T=0
280 FOR I=1 TO K
290 LET T=T+VAL(R$(I,5))
300 NEXT I
310 PRINT "EL DEPOSITO TOTAL ES DE ";T;" PTAS"
400 END
```

6. Añadir esto al programa:

```
400 REM ENCONTRAR LOS NOMBRES DE LOS CLIENTES CUYOS
410 REM DEPOSITOS ESTEN ENTRE 10000 Y 20000 PESETAS
420 FOR I=1 TO K
430 IF VAL(R$(I,5))>=10000 AND VAL(R$(I,5))<=20000 THEN
    PRINT "EL DEPOSITO DE ";R$(I,1);" ES DE ";R$(I,5);" PTAS"
440 NEXT I
500 END
```

7. Añadir esto al programa:

```
500 REM ENCONTRAR UN CLIENTE A PARTIR DE UN NUMERO
    DE SERIE DE UN COCHE
510 INPUT "ENTRE EL NUMERO DE SERIE DEL COCHE
    (O CERO PARA ACABAR) ";NS$
515 IF SN$="0" THEN GOTO 610
520 FOR I=1 TO K
530 IF SN$<>R$(I,4) THEN GOTO 590
540 PRINT "EL COCHE FUE ALQUILADO A ..."
550 PRINT "NOMBRE: ";R$(I,1)
560 PRINT "DIRECCION: ";R$(I,2)
570 PRINT "TELEFONO: ";R$(I,3)
580 PRINT "DEPOSITO: ";R$(I,5):LET I=K
590 NEXT I
600 GOTO 510
610 END
```

Capítulo 9

Ejercicios 9.1

1. El efecto sería que se produjeran nombres del tipo "ASDECORAZONES" en vez de "AS DE CORAZONES" y se almacenasen en C$.

2.
```
100 REM PRIMEROS NOMBRES
110 DATA "MERCEDES", "CRISTINA", "AMELIA", "CARLOTA"
120 REM SEGUNDOS NOMBRES
130 DATA "ANA", "DANIELA", "MARIA", "MANUELA"
140 REM
150 DIM F$(4), M$(4), N$(16)
160 REM MODULO DE ASIGNACION
170 FOR K = 1 TO 4
180 READ F$(K)
190 NEXT K
200 FOR K=1 TO 4
210 READ M$(K)
220 NEXT K
230 REM
240 REM MODULO DE CALCULO
250 FOR J=1 TO 4
260 FOR K=1 TO 4
270 LET N$(C)=F$(J)+" "+M$(K)+" "+"SEÑORITA"
280 NEXT K
```

```
290 NEXT J
300 REM
310 REM MODULO DE SALIDA
320 FOR K = 1 TO 16
330 PRINT N$(K)
340 NEXT K
350 END
```

```
]RUN
MERCEDES ANA SEÑORITA
MERCEDES DANIELA SEÑORITA
MERCEDES MARIA SEÑORITA
MERCEDES MANUELA SEÑORITA
CRISTINA ANA SEÑORITA
CRISTINA DANIELA SEÑORITA
CRISTINA MARIA SEÑORITA
CRISTINA MANUELA SEÑORITA
AMELIA ANA SEÑORITA
AMELIA DANIELA SEÑORITA
AMELIA MARIA SEÑORITA
AMELIA MANUELA SEÑORITA
CARLOTA ANA SEÑORITA
CARLOTA DANIELA SEÑORITA
CARLOTA MARIA SEÑORITA
CARLOTA MANUELA SEÑORITA
```

3.

```
100 REM MODULO DE ASIGNACION
110 REM
120 INPUT "ESCRIBA UNA PALABRA ";W$
130 REM
140 REM MODULO DE CALCULO
150 LET A$ = LEFT$(W$,1)
160 LET B$ = MID$(W$,2)
170 LET W$ = B$ + A$ + "AY"
180 REM
190 REM MODULO DE SALIDA
200 REM
210 PRINT "ESA PALABRA EN LATIN PARDO ES ";W$
220 END
```

4.

```
100 REM MODULO DE SALIDA
110 REM
120 PRINT "INTRODUZCA NOMBRE Y APELLIDO."
130 "SEPARADOS POR UN ESPACIO: ";N$
140 REM
150 REM MODULO DE CALCULO
160 REM BUSCA EL ESPACIO
170 FOR K = 1 TO LEN(N$)
180 IF MID$(N$,K,1) = "  " THEN GOTO 210
190 NEXT K
200 PRINT "NO HAY ESPACIO, PRUEBE OTRA VEZ." :GOTO 120
210 REM
220 REM SEPARAR Y REORDENAR NOMBRES
230 LET F$ = LEFT$(N$,K - 1)
240 LET L$ = MID$(N$,K + 1)
```

```
250 LET N$ = L$ + "   " + F$
260 REM
270 REM MODULO DE SALIDA
280 PRINT "SU APELLIDO Y SU NOMBRE SON: ";N$
290 END
```

Ejercicios 9.2

1.
```
10 INPUT "CUAL ES EL NUMERO? ";N
20 REM
30 REM STEPS 1 AND 2 COMBINADOS
40 LET N1$ = "00" + STR$(INT(100*N + .5))
50 REM
60 REM PASOS 3, 4, 5, Y 6 COMBINADOS
70 LET N$ = STR$(VAL(LEFT$(N1$,LEN(N1$) - 2))) + "." +
   RIGHT$(N1$,2)
80 REM
90 PRINT "HE AQUI EL NUMERO EN PESETAS Y CENTIMOS:" ;N$
100 END
```

Ejercicios 9.3

1.
```
10 REM MODULO DE ASIGNACION
20 INPUT "CUANTAS PERSONAS? ";N
30 DIM R$(N)
40 LET B$ = "          "
50 FOR I = 1 TO N
60 PRINT
70 INPUT "APELLIDO: ";A$
80 INPUT "NOMBRE: ";N$
90 INPUT "TELEFONO: ";T$
95 INPUT "SALARIO: ";S$
100 LET L$ = LEFT$(A$ + B$,10)
110 LET F$ = LEFT$(N$ + B$,9)
120 LET T$ = LEFT$(T$ + B$,8)
125 LET S$ = RIGHT$(B$ + S$,5)
130 LET R$(I) = A$ + N$ + T$ + S$
140 NEXT I
150 REM MODULO DE CALCULO, ORDENACION DE SALARIOS
160 FOR J = 1 TO N - 1
170 FOR K = 1 TO N - 1
180 REM EL CAMPO DEL SALARIO EMPIEZA EN LA POSICION 28
190 IF MID$(R$(K),28,5) > MID$(R$(K + 1),28,5) THEN
    LET X$ = R$(K) :LET R$(K) = R$(K + 1) :LET R$(K + 1) = X$
200 NEXT K
210 NEXT J
220 REM MODULO DE SALIDA
230 PRINT
240 FOR I = 1 TO N
250 PRINT R$(I)
260 NEXT I
270 END
```

2.
```
100 REM MODULO DE ASIGNACION
110 REM
120 INPUT "CUANTOS ENTEROS HAY QUE ORDENAR?"
```

```
130 DIM A$(N)
140 FOR K=1 TO N
150 PRINT "ESCRIBA EL NUMERO ";K;": ";
160 INPUT " ";NUM
170 REM AJUSTE A LA DERECHA EN UN CAMPO DE 10 CARACTERES
180 LET N$=RIGHT$("          "+STR$(NUM),10)
190 LET A$(K)=N$
200 NEXT K
210 REM
220 REM MODULO DE ORDENACION
230 FOR P=1 TO N-1
240 FOR C=1 TO N-1
250 IF A$(C)>A$(C+1) LET T$=A$(C) :
    LET A$(C)=A$(C+1) :LET A$(C+1)=T$
260 NEXT C
270 NEXT P
280 REM
290 REM MODULO DE SALIDA
300 FOR K=1 TO N
310 PRINT A$(K)
320 NEXT K
330 END
```

```
]RUN
CUANTOS ENTEROS HAY QUE ORDENAR? 5
ESCRIBA EL NUMERO 1: 0342
ESCRIBA EL NUMERO 2: 00013
ESCRIBA EL NUMERO 3: 24
ESCRIBA EL NUMERO 4: 92142
ESCRIBA EL NUMERO 5: 1
         1
        13
        24
       342
     92142
```

3.
```
100 REM MODULO DE ASIGNACION
110 REM
120 INPUT "CUANTAS PALABRAS HAY QUE ORDENAR? ";N
130 DIM A$(N)
140 FOR K=1 TO N
150 PRINT "ESCRIBA LA PALABRA ";K;": ";
160 INPUT "";W$
170 REM AJUSTE A LA IZQUIERDA EN UN CAMPO DE 15 CARACTERES
180 LET W$=LEFT$(W$+"                ",15)
190 LET A$(K)=W$
200 NEXT K
210 REM
220 REM MODULO DE ORDENACION
230 FOR P=1 TO N-1
240 FOR C=1 TO N-1
250 IF A$(C)>A$(C+1) THEN LET T$=A$(C) :
    LET A$(C)=A$(C+1) :LET A$(C+1)=T$
260 NEXT C
```

```
270 NEXT P
280 REM
290 REM MODULO DE SALIDA
300 PRINT
310 PRINT "AQUI ESTAN ORDENADAS Y AJUSTADAS A LA IZQUIERDA:"
320 FOR K=1 TO N
330 PRINT A$(K)
340 NEXT K
350 END
```

```
]RUN
CUANTAS PALABRAS HAY QUE ORDENAR? 4
ESCRIBA LA PALABRA 1: PERRO
ESCRIBA LA PALABRA 2: GATO
ESCRIBA LA PALABRA 3: ZAPATO
ESCRIBA LA PALABRA 4: COCHE
AQUI ESTAN ORDENADAS Y AJUSTADAS A LA IZQUIERDA
COCHE
GATO
PERRO
ZAPATO
```

4. Cambie sólo las líneas 180 y 310:

```
180 LET W$=RIGHT$("                    "+W$,15)
310 PRINT "AQUI ESTAN ORDENADAS Y AJUSTADAS A LA DERECHA:"
```

El resto del programa es el mismo.

Capítulo 10

Ejercicios 10.1

1.
```
REM A$(N,M) ES UNA MATRIZ TIPO CADENA
REM N ES EL NUMERO DE FILAS
REM M ES EL NUMERO DE COLUMNAS
FOR I=1 TO N
FOR J=1 TO M
INPUT "ENTRE UNA CADENA ";A$(I,J)
NEXT J
NEXT I
```

2.
```
REM A(N,M) ES LA MATRIZ
REM N ES EL NUMERO DE FILAS
REM M ES EL NUMERO DE COLUMNAS
REM S ( ) ES LA MATRIZ QUE CONTIENE LA SUMA DE CADA COLUMNA
FOR I=1 TO M
LET S(I)=0
FOR J=1 TO N
LET S(I)+A(J,I)
NEXT J
NEXT I
```

3.
```
255 REM ESTA ES LA CONTINUACION
260 REM DEL PROGRAMA DE LA COMPAÑIA DE TAXIS YELLOW STONE
270 DIM S(Q),T(Q)
280 FOR I=1 TO Q
290 LET S(I)=S(I)+VAL(MID$(R$(I),38,4))
300 LET T(I)=T(I)+VAL(RIGHT$(R$(I),4))
400 NEXT I
```

4.
```
 10 INPUT "ESCRIBA EL NUMERO DE CLIENTES EN
    LA LISTA ";N
 20 DIM R$(N)
 30 FOR I=1 TO N
 40 LET B$ ="
 50 INPUT "APELLIDO ";F1$
 60 INPUT "NOMBRE ";F2$
 70 INPUT "DOMICILIO ";F3$
 80 INPUT "CIUDAD ";F4$
 90 INPUT "DISTRITO ";F5$
100 INPUT "TELEFONO ";F6$
110 LET F1$ =LEFT$(F$ + B$,30)
120 LET F2$ =LEFT$(F$ + B$,15)
130 LET F3$ =LEFT$(F$ + B$,30)
140 LET F4$ =LEFT$(F$ + B$,20)
150 LET F5$ =LEFT$(F$ + B$,14)
160 LET F6$ =LEFT$(F$ + B$,5)
170 LET R$(I)=F1$ + F2$ + F3$ + F4$ + F5$ + F6$
180 NEXT I
```

5.
```
FOR I=1 TO Q
INPUT "INTRODUZCA CAMPO 1: ";F1$
INPUT "INTRODUZCA CAMPO 2: ";F2$
   :
etc.
   :
LET F1$ =RIGHT$(B$ + F1$,A1)
LET F2$ =RIGHT$(B$ + F2$,A2)
   :
etc.
   :
LET R$(I)=F1$ + F2$ + ... etc...
NEXT I
```

6.
```
REM ORDENACION DESCENDIENTE
REM N ES EL NUMERO DE ITEMS A ORDENAR
FOR K=1 TO N-1
FOR J=1 TO N-K
IF A(J)<A(J+1) THEN LET L=A(J) :
   LET A(J)=A(J+1) :LET A(J+1)=L
NEXT J
NEXT K
```

7.
```
260 REM ORDENACION SEGUN EL CAMPO MAS A LA IZQUIERDA
270 FOR K=1 TO Q-1
280 FOR J=1 TO Q-K
290 IF LEFT$(R$(J),25)>LEFT$(R$(J+I),25) THEN LET L$ =R$(J):
      LET R$(J)=R$(J+1) :LET R$(J+1)=L$
```

```
300 NEXT J
310 NEXT K
```

8.
```
180 REM CONTINUACION DEL PROGRAMA DEL PROBLEMA 4
190 PRINT "1. ORDENACION POR APELLIDOS"
200 PRINT "2. ORDENACION POR NOMBRES"
210 PRINT "3. ORDENACION POR DOMICILIOS"
220 PRINT "4. ORDENACION POR CIUDADES"
230 PRINT "5. ORDENACION POR DISTRITOS"
240 PRINT "6. ORDENACION POR TELEFONOS"
250 INPUT "ESCRIBA EL NUMERO DESEADO ";W
260 IF W=1 THEN LET C=1: D=30
270 IF W=2 THEN LET C=31: D=15
280 IF W=3 THEN LET C=46: D=30
290 IF W=4 THEN LET C=76: D=20
300 IF W=5 THEN LET C=96: D=5
310 IF W=6 THEN LET C=101: D=14
320 REM ORDENACION SEGUN EL CAMPO INDICADO
330 FOR I=1 TO N-1
335 FOR J=1 TO N-1
340 IF MID$(R$(J),C,D)<MID$(R$(J+1),C,D) THEN
    LET L$=R$(J): LET R$(J)=R$(J+1):
    LET R$(J+1)=L$
350 NEXT J
360 NEXT I
```

9.
```
REM R$(N,M) ES UNA MATRIZ BIDIMENSIONAL
REM A ES LA FILA DONDE REALIZAR LA BUSQUEDA
REM S$ ES LA CADENA BUSCADA
LET L=0
FOR I=1 TO N
IF S$=R$(I,A) THEN LET L=I :LET I=N
NEXT I
IF L=0 THEN (instrucción)
```

10. Los cambios siguientes lo harán:

```
270 INPUT "ENTRE EL TELEFONO BUSCADO ";S$
280 LET S$=LEFT$(S$+B$,8)
320 IF S$=RIGHT$(R$(I),8) THEN LET L=I :LET I=N
```

Ejercicios 10.2

1. Algoritmo V.

2. Algoritmo II.

3. Sustituya los módulos IV y V del programa Shady Lane por los siguientes:

```
270 REM MODULO DE ORDENACION POR EDAD
280 FOR I=1 TO N-1
290 FOR J=1 TO N-I
300 IF MID$(T$(J),26,3)>MID$(T$(J+1),26,3) THEN
    LET A$=T$(J) :LET T$(J)=T$(J+1) :LET T$(J+1)=A$
310 NEXT J
320 NEXT I
```

```
325 REM
330 REM MODULO DE LISTADO
340 PRINT "APELLIDO";TAB(17);"NOMBRE";TAB(35);"EDAD"
350 FOR I=1 TO N
360 PRINT LEFT$(T$(I),15);TAB(17);MID$(T$(I),16,10);
    TAB(35);MID$(T$(I),26)
370 NEXT I
375 REM
380 REM MODULO DE ORDENACION POR NOMBRE
390 FOR I=1 TO N-1
400 FOR J=1 TO N-I
410 IF MID$(T$(J),16,10)>MID$(T$(J+1),16,10) THEN
    LET A$=T$(J):LET T$(J)=T$(J+1):LET T$(J+1)=A$
420 NEXT J
430 NEXT I
440 REM
450 REM MODULO DE LISTADO
460 PRINT "APELLIDO";TAB(17);"NOMBRE";TAB(35);"EDAD"
470 FOR I=1 TO N
480 PRINT LEFT$(T$(I),15);TAB(17);MID$(T$(I),16,10);
    TAB(35);MID$(T$(I),26)
490 NEXT I
500 END
```

4. Sustituya el módulo IV de Shady Lane por:

```
270 REM MODULO IV "ORDENACION POR EDAD Y NOMBRE"
280 FOR I=1 TO N-1
290 FOR J=1 TO N-1
300 LET H1$=MID$(T$(J),26,3)+LEFT$(T$,15)
310 LET H2$=MID$(T$(J+1),26,3)+LEFT$(T$(J+1),15)
320 IF H1$>H2$ THEN LET L$=T$(J):LET T$(J)=T$(J+1):
    LET T$(J+1)=A$
330 NEXT J
340 NEXT I
350 REM FIN DEL MODULO IV
```

5.

```
 10 REM MODULO I - CREACION DE REGISTROS
 20 INPUT "CUANTOS LIBROS HAY? ";N
 30 DIM G$(N)
 40 LET B$="                                        "
 45 FOR I=1 TO N
 50 INPUT "APELLIDO DEL AUTOR: ";A$
 60 INPUT "NOMBRE DEL AUTOR: ";N$
 70 INPUT "TITULO: ";T$
 80 INPUT "EDITORIAL: ";E$
 90 INPUT "PRECIO: ";P$
100 LET G$(I)=LEFT$(A$+B$,25)+LEFT$(N$+B$,15)+
    LEFT$(T$+B$,30)+LEFT$(E$+B$,25)+RIGHT$(B$+P$,3)
110 NEXT I
120 REM MODULO II - ORDENACION Y LISTADOS
125 REM ORDENACION POR APELLIDOS
130 FOR I=1 TO N-1
140 FOR J=1 TO N-1
150 IF LEFT$(G$(J),25)>LEFT$(G$(J+1),25) THEN
    LET A$=G$(J):LET G$(J)=G$(J+1):LET G$(J+1)=A$
```

```
160 NEXT J
170 NEXT I
180 PRINT "ORDENADOS POR APELLIDOS DEL AUTOR"
190 PRINT
200 FOR I=1 TO N
210 PRINT G$(I)
220 NEXT I
225 REM ORDENACION POR TITULOS
230 FOR I=1 TO N-1
240 FOR J=1 TO N-1
250 IF MID$(G$(J),41,30)>MID$(G$(J+1),41,30) THEN
    LET A$=G$(J) : LET G$(J)=G$(J+1) :LET G$(J+1)=A$
260 NEXT J
270 NEXT I
280 PRINT "ORDENADOS POR TITULOS"
290 PRINT
300 FOR I=1 TO N
310 PRINT G$(I)
320 NEXT I
330 REM MODULO III - COMPARACION DE PRECIOS
340 INPUT "INTRODUZCA UN PRECIO PARA COMPARAR: ";X
350 FOR I=1 TO N
360 IF X>VAL(RIGHT$(G$(I),3)) THEN PRINT G$(I)
370 NEXT I
380 END
```

6. Reemplace el módulo III del programa del *Opera Times* por el siguiente:

```
160 REM MODULO III - CALCULO DEL NUMERO DE LINEAS Y PRECIO
165 LET N=INT(L/44)
170 LET C=5500*N
175 IF L=44*N THEN GOTO 190
180 IF L-44*N<=22 THEN C=C+2800
185 IF L-44*N>22 THEN C=C+5500
190 REM FIN DEL MODULO III
```

7. Reemplace el módulo III por:

```
160 REM MODULO III
165 LET N=INT(L/44)
170 IF L=44*N THEN LET S=N
175 IF L>44*N THEN LET S=N+1
185 IF S>4 THEN LET C=5500*4+4500*(S-4):GOTO 195
190 LET C=5500*S
195 REM FIN DEL MODULO III
```

Ejercicios 10.3

1. Cambie la línea 240 del módulo II

```
240 LET Q=25
```

Ejercicios 10.4

1. Añada estas líneas:

```
 70 ON K GOSUB 90,190,260,400
190 MODULO II - BUSQUEDA DE ESTUDIANTES POR EL NUMERO
```

```
200 INPUT "INTRODUZCA EL NUMERO DE MATRICULA: ";Q$
210 LET K=1
220 GOSUB 330
230 RETURN
250 REM
260 REM MODULO III - BUSQUEDA POR APELLIDOS
270 INPUT "ENTRE LOS APELLIDOS: ";Q$
280 LET K=2
290 GOSUB 330
295 RETURN
330 REM SUBRUTINA DE BUSQUEDA
340 LET L=0
350 FOR I=1 TO N
360 IF Q$=S$(I,K) THEN LET L=I
370 NEXT I
380 IF <>0 THEN PRINTS$ (L,1),S$(L,2),S$(L,3),S$(L,4)
390 RETURN
400 END
```

2. Cambie la línea 290 del módulo III:

```
290 IF Q$=S$(I,2) THEN PRINT S$(L,1),S$(L,2),S$(L,3),S$(L,4)
```

y borre la 310.

Capítulo 11

Ejercicios 11.1

1.
```
10 REM MODULO I - ENTRADA DE DATOS
20 INPUT "CUANTOS NOMBRES VA A INTRODUCIR? ";N
30 DIM T$(N)
35 LET B$="                                                    "
40 FOR I=1 TO N
45 INPUT "APELLIDO: ";A$
50 INPUT "NOMBRE: ";N$
60 INPUT "DOMICILIO: ";D$
70 INPUT "CIUDAD: ";C$
80 INPUT "DISTRITO: ";Z$
90 INPUT "TELEFONO: ";T$
100 LET A$=LEFT$(A$+B$,15)
110 LAT N$=LEFT$(N$+B$,10)
120 LET D$=LEFT$(D$+B$,35)
130 LET C$=LEFT$(C$+B$,15)
140 LET Z$=LEFT$(Z$+B$,5)
150 LET T$=LEFT$(T$+B$,14)
160 LET T$(I)=A$+N$+D$+C$+Z$+T$
170 NEXT I
180 REM MODULO II - ALMACENAMIENTO DE UNA FILA EN UN DISCO
190 PRINT CHR$(4);"OPEN DIRECTORIO"
200 PRINT CHR$(4);"WRITE DIRECTORIO"
210 FOR I=1 TO N
220 PRINT T$(I)
230 NEXT I
240 PRINT CHR$(4);"CLOSE DIRECTORIO"
250 END
```

2.
```
100 REM MODULO I
110 INPUT "CUANTOS COCHES HAY? ";N
120 DIM C$(N)
130 LET B$ = "                    "
140 FOR I = 1 TO N
150 INPUT "TIPO: ";V$
160 INPUT "FABRICANTE: ";M$
170 INPUT "MODELO: ";K$
180 INPUT "AÑO: ";Y$
190 INPUT "KILOMETRAJE: ";I$
200 INPUT "PRECIO DIARIO: ";P$
210 INPUT "PRECIO POR KILOMETRO: ";A$
220 LET V$ = LEFT$(V$ + B$,15)
230 LET M$ = LEFT$(M$ + B$,15)
240 LET K$ = LEFT$(K$ + B$,15)
250 LET Y$ = LEFT$(Y$ + B$,4)
260 LET I$ = RIGHT$(B$ + I$,6)
270 LET P$ = RIGHT$(B$ + P$,5)
280 LET A$ = RIGHT$(B$ + A$,5)
290 LET C$(I) = V$ + M$ + K$ + Y$ + I$ + P$ + A$
300 NEXT I
310 REM MODULO II
320 PRINT CHR$(4);"OPEN RAL"
330 PRINT CHR$(4);"WRITE RAL"
340 FOR I = 1 TO N
350 PRINT C$(I)
360 NEXT I
370 PRINT CHR$(4);"CLOSE RAL"
380 END
```

3.
```
100 REM MODULO I
110 INPUT "CUANTOS ESTUDIANTES HAY? ";N
120 DIM S$(N)
130 LET B$ = "                    "
140 FOR I = 1 TO N
150 PRINT "ENTRADA ESTUDIANTE ";I
160 INPUT "APELLIDO: ";A$
170 INPUT "NOMBRE: ";N$
180 INPUT "MATRICULA: ";M$
190 INPUT "PRIMER PARCIAL: ";G1$
200 INPUT "SEGUNDO PARCIAL: ";G2$
210 INPUT "TERCER PARCIAL: ";G3$
220 INPUT "EXAMEN FINAL: ";E$
230 LET A$ = LEFT$(A$ + B$,15)
240 LET N$ = LEFT$(N$ + B$,10)
250 LET M$ = RIGHT$(B$ + M$,7)
260 LET G1$ = RIGHT(B$ + G1$,3)
270 LET G2$ = RIGHT$(B$ + G2$,3)
280 LET G3$ = RIGHT$(B$ + G3$,3)
290 LET E$ = RIGHT(B$ + E$,3)
300 LET S$(I) = A$ + N$ + M$ + G1$ + G2$ + G3$ + E$
310 NEXT I
320 REM MODULO II
330 PRINT CHR$(4);"OPEN CALIFICACIONES"
340 PRINT CHR$(4);"WRITE CALIFICACIONES"
350 FOR I = 1 TO N
```

```
360 PRINT S$(I)
370 NEXT I
380 PRINT CHR$(4);"CLOSE CALIFICACIONES"
390 END
```

Ejercicios 11.2

1.
```
190 REM MODULO DE ORDENACION DE REGISTROS POR APELLIDOS
192 FOR J=1 TO I-1
194 FOR K=1 TO I-J
196 IF LEFT$(T$(K),15)>LEFT$(T$(K+1),15) THEN
    LET A$=T$(K) :LET T$(K)=T$(K+1) :LET T$(K+1)=A$
197 NEXT K
198 NEXT J
```

2. *a)*
```
100 REM LECTURA DEL DISCO DEL LISTIN TELEFONICO
100 DIM T$(100) :REM ESTO SUPONE QUE HAY MENOS DE 100 NOMBRES
120 ONERR GOTO 190
130 PRINT CHR$(4);"OPEN DIRECTORIO"
140 PRINT CHR$(4);"READ DIRECTORIO"
150 LET N=0
160 LET N=N+1
170 INPUT T$(N)
180 GOTO 160
190 PRINT CHR$(4);"CLOSE DIRECTORIO"
195 POKE 216,0
200 LET N=N-1 :REM N ES EL NUMERO DE REGISTROS EN LA FILA
210 END
```

Si se supiera el número de registros en la fila, el programa quedaría así:

```
100 INPUT "CUANTOS REGISTROS HAY EN LA FILA? ";N
110 DIM T$(N)
120 PRINT CHR$(4);"OPEN DIRECTORIO"
130 PRINT CHR$(4);"READ DIRECTORIO"
140 FOR I=1 TO N
150 INPUT T$(I)
160 NEXT I
170 PRINT CHR$(4);"CLOSE DIRECTORIO"
180 END
```

b)
```
100 REM LECTURA DE LA FILA RAL
110 DIM C$(100) :REM ESTO SUPONE QUE HAY MENOS DE 100 VEHICULOS
120 ONERR GOTO 190
130 PRINT CHR$(4);"OPEN RAL"
140 PRINT CHR$(4);"READ RAL"
150 LET N=0
160 LET N=N+1
170 INPUT C$(N)
180 GOTO 160
190 PRINT CHR$(4);"CLOSE RAL"
195 POKE 216,0
200 LET N=N-1 :REM N ES EL NUMERO DE REGISTROS EN LA FILA
210 END
```

c)
```
100 REM LECTURA DE LA FILA GRADE
110 DIM S$(100) :REM ESTO SUPONE QUE HAY MENOS DE 100 ESTUDIANTES
120 ONERR GOTO 190
```

```
130 PRINT CHR$(4);"OPEN CALIFICACIONES"
140 PRINT CHR$(4);"READ CALIFICACIONES"
150 LET N=0
160 LET N=N+1
170 INPUT S$(N)
180 GOTO 160
190 PRINT CHR$(4);"CLOSE CALIFICACIONES"
195 POKE 216,0
200 LET N=N-1 :REM N ES EL NUMERO DE REGISTROS EN LA FILA
210 END
```

Ejercicios 11.3

1. *a)*

```
10 REM MODULO I - ENTRADA DE DATOS EN LA MEMORIA
20 INPUT "CUANTOS NOMBRES VA A INTRODUCIR? ";N
30 DIM T$(N,6)
35 LET B$="                                                    "
40 FOR I=1 TO N
45 INPUT "APELLIDO: ";A$
50 INPUT "NOMBRE: ";N$
60 INPUT "DOMICILIO: ";D$
70 INPUT "CIUDAD: ";C$
80 INPUT "CODIGO: ";Z$
90 INPUT "TELEFONO: ";T$
100 LET T$(I,1)=LEFT$(A$+B$,15)
110 LET T$(I,2)=LEFT$(N$+B$,10)
120 LET T$(I,3)=LEFT$(D$+B$,35)
130 LET T$(I,4)=LEFT$(C$+B$,15)
140 LET T$(I,5)=LEFT$(Z$+B$,5)
150 LET T$(I,6)=LEFT$(T$+B$,14)
170 NEXT I
180 REM MODULO II - ALMACENAMIENTO DE LA FILA EN UN DISCO
190 PRINT CHR$(4);"OPEN DIRECTORIO"
200 PRINT CHR$(4);"WRITE DIRECTORIO"
210 FOR I=1 TO N
215 FOR K=1 TO 6
220 PRINT T$(I,K)
225 NEXT K
230 NEXT I
240 PRINT CHR$(4);"CLOSE DIRECTORIO"
250 END
```

b)

```
100 REM MODULO I
110 INPUT "CUANTOS VEHICULOS HAY? ";N
120 DIM C$(N,7)
130 LET B$="                                                    "
140 FOR I=1 TO N
150 INPUT "TIPO DEL VEHICULO: ";V$
160 INPUT "FABRICANTE: ";M$
170 INPUT "MODELO: ";K$
180 INPUT "AÑO: ";Y$
190 INPUT "KILOMETRAJE: ";I$
200 INPUT "PRECIO POR DIA: ";D$
210 INPUT "PRECIO POR KILOMETRO: ";A$
220 LET C$(I,1)=LEFT$(V$+B$,15)
```

```
230 LET C$(I,2)=LEFT$(M$+B$,15)
240 LET C$(I,3)=LEFT$(K$+B$,15)
250 LET C$(I,4)=LEFT$(Y$+B$,4)
260 LET C$(I,5)=RIGHT$(B$+I$,6)
270 LET C$(I,6)=RIGHT$(B$+P$,5)
280 LET C$(I,7)=RIGHT$(B$+A$,5)
300 NEXT I
310 REM MODULO II
320 PRINT CHR$(4);"OPEN RAL"
330 PRINT CHR$(4);"WRITE RAL"
340 FOR I=1 TO N
345 FOR K=1 TO 7
350 PRINT C$(I,K)
355 NEXT K
360 NEXT I
370 PRINT CHR$(4);"CLOSE RAL"
380 END
```

c)
```
100 REM MODULO I
110 INPUT "CUANTOS ESTUDIANTES HAY? ";N
120 DIM S$(N,7)
130 LET B$="                    "
140 FOR I=1 TO N
150 PRINT "ENTRADA ESTUDIANTE ";I
160 INPUT "APELLIDO: ";A$
170 INPUT "NOMBRE: ";N$
180 INPUT "MATRICULA: ";M$
190 INPUT "PRIMER PARCIAL: ";G1$
200 INPUT "SEGUNDO PARCIAL: ";G2$
210 INPUT "TERCER PARCIAL: ";G3$
220 INPUT "EXAMEN FINAL: ";E$
230 LET S$(I,1)=LEFT$(A$+B$,15)
240 LET S$(I,2)=LEFT$(N$+B$,10)
250 LET S$(I,3)=RIGHT$(B$+M$,7)
260 LET S$(I,4)=RIGHT$(B$+G1$,3)
270 LET S$(I,5)=RIGHT$(B$+G2$,3)
280 LET S$(I,6)=RIGHT$(B$+G3$,3)
290 LET S$(I,7)=RIGHT$(B$+E$,3)
310 NEXT I
320 REM MODULO II
330 PRINT CHR$(4);"OPEN CALIFICACIONES"
340 PRINT CHR$(4);"WRITE CALIFICACIONES"
350 FOR I=1 TO N
355 FOR K=1 TO 7
360 PRINT S$(1,K)
365 NEXT K
370 NEXT I
380 PRINT CHR$(4);"CLOSE CALIFICACIONES"
390 END
```

Ejercicios 11.4

1. Para cambiar los programas escritos en el ejercicio 11.1, de tal manera que nuevos registros puedan ser añadidos a los ya existentes, simplemente modifique dos líneas en cada programa:

a) 20 INPUT "CUANTOS NOMBRES NUEVOS VA A INTRODUCIR? ";N
190 PRINT CHR$(4);"APPEND DIRECTORIO"

b) 110 INPUT "CUANTOS VEHICULOS NUEVOS HAY? ";N
320 PRINT CHR$(4);"APPEND RAL"

c) 110 INPUT "CUANTOS ESTUDIANTES NUEVOS HAY? ";N
330 PRINT CHR$(4);"APPEND CALIFICACIONES"

Indice alfabético

ABS(X), 52.
ACS, método de módulos, 87, 88, 93.
Acumulador, 119.
Ajustado, proceso de, 224, 226.
Algoritmo, 229, 230.
Almacenamiento temporal, 27.
Altavoz, 302, 305.
AND, 138, 150.
Anidamiento de lazos, 185.
Archivo secuencial, 269, 273.
Aritmética
  orden operacional, 33.
  práctica, 46.
  símbolos, 31.
Argumento, 49, 50.
ASC(X$), 313.
ASCII, código, 313.
Asignación, 21, 22, 32.
ATN(X), 55.

BAD SUBSCRIPT ERROR, 324.
Bit, 282.
Booting, 318.
Bucle, 153, 154, 155, 170.
Buffer, 271.
Bugs, 323.
Byte, 282.

Cadena
  datos de, 17.
  función de, 48, 57.
  matrices de, 197.
  variable de, 20.
CALL, 281, 285, 293.
Campo, 217, 224, 275.
Campo clave, 235.
CANT'T CONTINUE ERROR, 327.
CATALOG, 319.
Cebar el sistema, 318.
CHR$(X), 315.
Clave de búsqueda, 200.
CLEAR, 80.
Código spaghetti, 259.
COLOR, 295, 322.
Copia, 24.
Coma, 101, 109.
Comillas, 22.
Concatenación
  definición de, 213.
  práctica, 223.
Conectivas lógicas, 138.
CONT, 327.
Contenido
  intercambio del, 25.
  práctica, 44.
Control del cursor, 329.

COS(X), 55.
CTRL-C, tecla de, 297, 331.
CTRL-X, 331.
Cuadrado mágico, 203.
Cursor, 106, 329, 330.

DATA, 115, 126.
Datos
  almacenamiento, 275.
  definición de, 17.
  tipos de, 17.
Debugging, 323.
Decisión, instrucción de, 135.
DEF FN, 307.
DEL, 92.
DELETE, 320.
Depuración de programas, 323, 327.
Desviación típica, 125, 180.
Diagrama de cajas, 18.
Dígito, 19.
DIMensión, 179.
DISK FULL, 320.
Diskette, 269, 317, 318, 319.
División, 31, 46.
DIVISION BY ZERO ERROR, 326.
Documentación, 259.
DOS (Disk Operating System), 317.

Edición, 329.
END, 78.
Enlazado, 153.
ESC, tecla, 330.
Exponenciación, 31, 46.
EXP(X), 55.
EXTRA IGNORED, 123.

Flag, 142.
FLASH, 289.
Flecha derecha, 330.
Flecha izquierda, 330.
FORT-NEXT, 153, 163, 167, 170.
Funciones
  aritméticas, 49, 50, 55.
  de cadena, 49, 57.
  definición de, 49.
  INT(X), 53, 62.
  LEFT$(X$,N), 60, 72.
  LEN(X$), 58, 67.
  longitud, 58.
  MID$(X$,J,K), 59, 70.
  RIGHT$(X$,N), 60, 71.
  RND(X), 54, 64.
  SQR(X), 50.
  subcadena, 59.
  STR$(X), 58, 69.
  TAB(X), 106, 112.
  tipos de, 49.
  valor, 58.

GET, 310.
GOSUB, 250, 262.
GOTO, 142.
GR, 295.
Gráficas
  de alta resolución, 295.
  de baja resolución, 299.

HCOLOR, 299, 322.
HGR, HGR2, 299.
HIMEN, 322.
Hir-res, 295.
HLIN C1,C2 AT R, 297.
HOME, 286.
HPLOT C,R, 299.
HPLOT C1, R1 TO C2,R2, 300.
HPLOT TO C,R, 301.
HTAB, 286, 287.

Indice, 176.
IF-THEN, 135, 149.
ILEGAL QUANTITY ERROR, 324.
Impresión
  coma, 101, 109.
  punto y coma, 103, 111.
  zonas de, 109.
Indicador, 122.
INIT, 97, 318.
INPUT, 122, 130.
Instrucciones
  de asignación, 23.
  de decisión, 135.
  de ramificación, 135, 151.
  múltiples, 86, 139.
INVERSE, 289.
Item, 102, 103, 129.

K, 283.

Lazo, 153, 170.
Lazos anidados, 172.
Lenguaje máquina, 293.
LET
  definición de, 21.
  práctica, 39.
  normas, 25.
Líneas numeradas, 76, 77.
LIST, 78, 91.
LOAD, 320.
LOG(X), 55.
LOMEN, 322.
Looping, 153.
Lo-res, 295.

Mandos exteriores, 302.
Matrices
  de cadena, 197.
  definición de, 175.
  normas de uso, 176.
  numéricas bidimensionales, 192, 208.
  numéricas unidimensionales, 176, 205.
  tipos de, 175.
Memoria, 269, 282.
  posición de, 282.
Mensaje demandante, 131.
Menú, 256, 257, 258.
Miniprocesador de Texto, 221, 222.
Modo de programación
  diferido, 75, 89.
  estructurada, 87, 229.
  inmediato, 75, 89.
Módulo
  de asignación, 87, 148.
  de cálculo, 87, 148.
  definición de, 241.
  de salida, 87, 148.
monitor, 285.
Multiplicación, 31, 46.

NEW, 80.
NEXT WOTHOUT FOR ERROR, 325.
NORMAL, 289.
Número áureo, 169.

OPEN, 177.
Operaciones relacionales, 137.
ONERR, 273, 274.
ON GOSUB, 256, 263.
ON GOTO, 256, 263.
OR, 138, 150.
Orden, 40, 89.
Ordenamiento, 183.
OUT OF DATA ERROR, 325.
OUT OF MEMORY ERROR, 326.

Paddle, 302.
Palabra reservada, 19.
PEEK, 281, 283.
Pixel, 299.
PLOT C,R, 296.
POKE, 282, 284, 291, 292.
Power, 317.
PRINT
  definición de, 29.
  práctica, 40, 109, 111.
  normas de uso, 30.
Programación
  deductiva (top-down), 258.
  definición de, 75.
  tipos de, 75, 87, 89, 229.
Pulsadores, 302.

RAM, 283.
Rango, 54.
READ, 115, 126.
REDIM'D ARRAY ERROR, 325.
Registro, 217, 224, 269, 273, 277.
Rellenado, proceso de, 224, 225, 227.
REMark, 85.
Resta, 31, 46.
RESET, tecla de, 297, 317.
RESTORE, 121, 129.
RETURN, 250, 262.
RETURN, tecla de, 39, 40.
RETURN WITHOUT GOSUB ERROR, 324.
ROM, 283.
ROT, 322.
RUN, 78.

Salto condicional, 143.
SAVE, 320.
SCALE, 322.
SCRN, 322.
Series de Taylor, 61.
Serie de Fibonacci, 168.
SIN(X), 55.
SPC, 322.

SPEED, 290.
STOP, 327.
STRING TOO LONG ERROR, 326.
Subrutina, 250, 263.
Suma, 31, 46.
Suma reiterativa, 119.
SYNTAX ERROR, 323.

Tabla, 192.
TAN(X), 55.
Tecla de repetición, 330.
TEXT, 335.
Transferencia condicional, 263.
TYPE MISMATCH ERROR, 324.
UNDEF'D FUNCTION ERROR, 326.
UNDEF'D STATEMENT ERROR, 324.

Valor absoluto, 52.
Variables
  de cadena, 20.
  indexadas, 205, 208.
  práctica, 42.
  numéricas, 19.
Ventana de texto, 291.
VLIN R1,R2 AT C, 297.
VTAB, 286, 287.

PROGRAMACION EN BASIC: UN METODO PRACTICO permite, de forma progresiva, evolucionar desde los conceptos básicos del lenguaje BASIC hasta las técnicas más sofisticadas de programación estructurada y tratamiento de archivos de datos. Imprescindible para el aprendizaje autodidáctico y de gran utilidad como guía en un curso de programación, los aciertos didácticos de su estructura —Diagramas de Cajas, Proyectos de Investigación, estudio en profundidad del diseño de algoritmos, entre otros— hacen de esta obra un manual de sumo interés para todos los que se adentran en el mundo de la Informática.

ANAYA MULTIMEDIA

Programación en BASIC

Howard Dachslager
Masato Hayashi
Richard Zucker

AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/

[FRA] Ce document a été préservé numériquement à des fins éducatives et d'études, et non commerciales.
[ENG] This document has been digitally preserved for educational and study purposes, not for commercial purposes.
[ESP] Este documento se ha conservado digitalmente con fines educativos y de estudio, no con fines comerciales.